Redbooks
ibm.com/redbooks

IBM zPDT Sysplex Extensions - 2020

Frank Kyne

Bill Ogden

IBM

Redbooks

IBM Redbooks

IBM zPDT Sysplex Extensions - 2020

October 2020

SG24-8386-02

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

Third Edition (October 2020)

This edition applies to the May 2020 Edition of the Application Developer Controlled Distribution (ADCD).

Contents

Notices

Trademarks

Preface

This IBM® Redbooks® publication describes the IBM System z® Personal Development Tool (IBM zPDT®) Sysplex Extensions 2020, 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 book is a significant revision of *zPDT 2017 Sysplex Extensions* , SG24-8386.

This package is designed and tested to be installed on top of a standard Application Developer Controlled Distribution (ADCD) environment. It provides the extra files that you need to create a two-way data sharing IBM z/OS® 2.4 sysplex that runs under IBM z/VM® in a zPDT environment.

Restriction: This package assumes that the base ADCD release is the May 2020 edition. If you plan to install the Sysplex Extensions on an older release, see *zPDT 2016 Sysplex Extensions*, SG24-8315, and its associated downloads.

This package differs from the zPDT sysplex package delivered in *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205, 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 Z® Development and Test Environment (previously known as RD&T), it might be used to reduce the effort to create a fully functional sysplex under zD&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 CPC.

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

Authors

This book was produced in the IBM Redbooks Poughkeepsie center.

Frank Kyne is the editor of Cheryl Watson™'s Tuning Letter. Before joining Watson and Walker, Frank was a project leader in the IBM Redbooks publication group in Poughkeepsie for 15 years, responsible for 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 IBM 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.

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

Stephen Anania
IBM USA

Andy Clifton
IBM UK

Alan Murphy
Watson & Walker, Ireland

Keith VanBeschoten
IBM USA

Now you can become a published author, too!

Comments welcome

Stay connected to IBM Redbooks

Summary of changes

This section describes the technical changes made in this edition of the book compared to the previous edition of this document, *IBM zPDT 2017 Sysplex Extensions*, SG24-8386. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-8386-02
for IBM zPDT Sysplex Extensions - 2020
as created or updated on October 16, 2020.

October 2020, Third Edition

This revision includes the following new and changed information.

New information

- This edition is based on z/OS 2.4, IBM CICS® TS 5.5, and IBM Db2® V12.
- The Sysplex Extensions 2020 environment was built and tested on zPDT GA10. GA10 delivers zPDT support for the new functions delivered in IBM z15™ CPCs. For more information about the various zPDT releases and the z System architecture level they support, see the Introduction chapter of *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.
- The Sysplex Extensions 2020 was built and tested on an early test level of the z/VM 7.1 ADCD.

January 2018, Second Edition

This revision includes the following new and changed information.

New information

- The installation chapter has extensive updates to reflect the ADCD Level change (Z22C to Z22D), volume serial number changes, and changes to data set names, particularly for the zFS data sets.
- We added sample jobs to help you clean up old IBM DB2® archive log files.
- The CICS CSD was updated in support of service updates to CICS TS 5.2.

Changed information

- This book assumes that you are doing an install ‘from scratch’. That it, it does not get into detailed discussions about migrating from previous levels. This is consistent with the methodology used by ADCD. However, it does provide additional information that might be valuable for readers that have an existing ADCD environment that they want to carry forward to the May 2017 level.

1 Introduction

This chapter introduces the concepts of both a base sysplex and a Parallel Sysplex for readers who might not be familiar with these environments. In particular, it describes the IBM System z Personal Development Tool (zPDT) and Application Developer Controlled Distribution (ADCD) support for a base sysplex and a Parallel Sysplex.

This chapter also includes a section that describes the remaining chapters of this book.

This chapter includes the following topics:

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 one cash register breaks 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 breaks down, you move and join the *end* of the other queue.You do not have to return all of your groceries, but 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 wonder about 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 sysplex control data sets, called *couple data sets* (CDSs). z/OS provides a software component, called *cross-system coupling facility* (XCF), that makes it easier for peer programs in the sysplex to communicate with each other.

The XCF also provides management and monitoring capabilities, which allows the operator to control the sysplex, and performance analysts and capacity planners to ensure that performance objectives are achieved.

Figure 1-1 shows the fundamental components of a base sysplex.

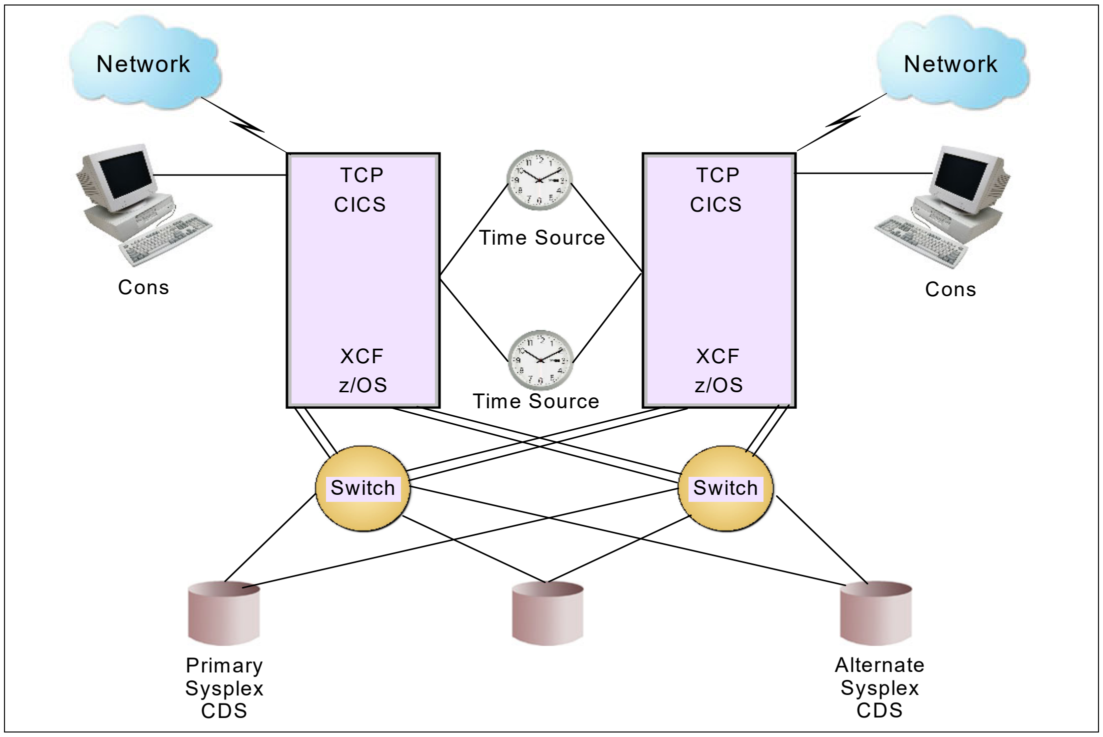


Figure 1-1 Components of a base sysplex

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 support data sharing with full data integrity and minimal overhead, and 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.

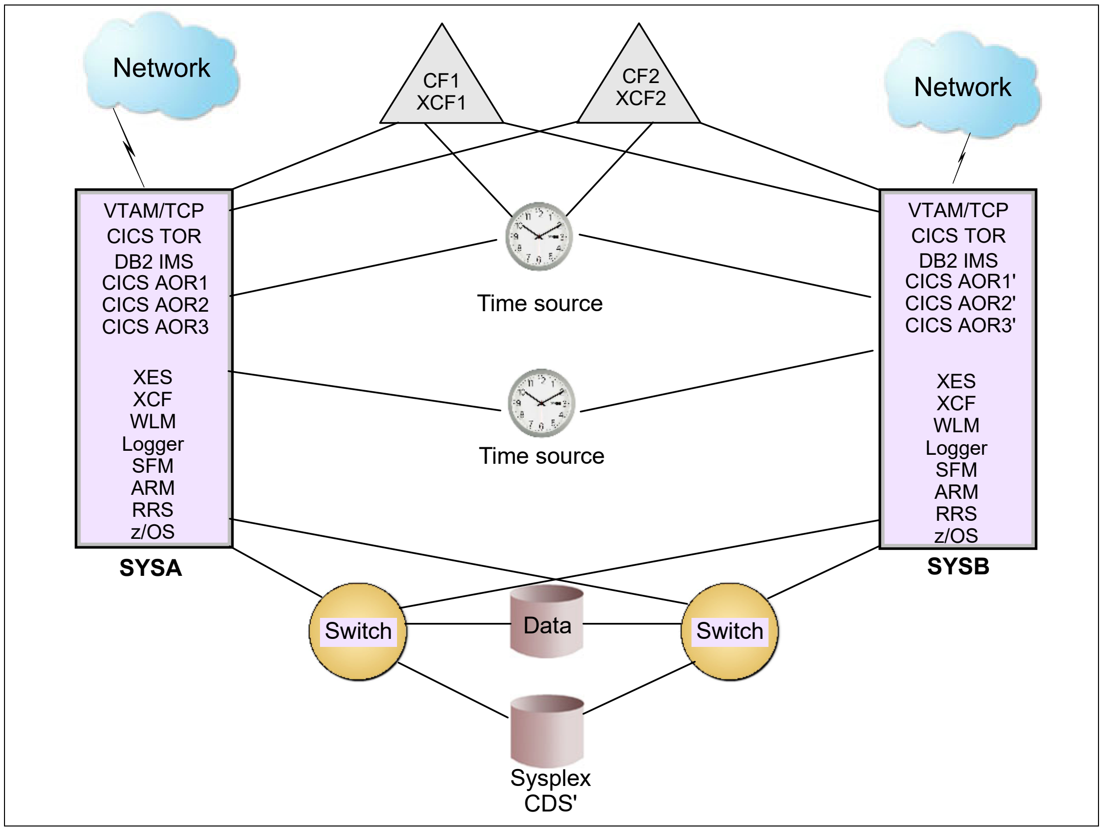


Figure 1-2 Components of a Parallel Sysplex

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

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

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

In a Parallel Sysplex environment, it is generally 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.

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 line. 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 checkout." If you do not get a response within a few seconds, you abandon that purchase and move on 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 a Parallel Sysplex (high availability and scalability) are more relevant than ever.

For software developers, a product that uses the capabilities of a 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 manual 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 their own. These companies requested extensions to the previously available ADCD sysplex download that provide working examples of sysplex-specific functions, such as the 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 you want it to work well in a Parallel Sysplex environment. To ensure that the application does not perform these actions, it must be tested in a multi-system Parallel Sysplex environment.

This package enables 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, the base package provided can deliver a significant time savings and encourage more customers and vendors to extend their support and use of a Parallel Sysplex.

1.4 Who is this deliverable aimed at?

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

The first target is the software developer whose product runs in a 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 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 who use zPDT to develop programs for a production Parallel Sysplex environment. Before moving your application to the production environment, you want to test it to ensure that the program behaves as expected. For example, it should not have any affinities to specific regions or subsystems, or to other transactions.

One of the objectives of a 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 running on a real IBM Z mainframe does; therefore, pre-production stress or “acceptance” testing should be carried out on a real mainframe.

Another aspect of resiliency is the ability to quickly and successfully recover from a failure. The failure might affect a single address space, an entire system, or possibly a coupling facility. You can use the Sysplex Extensions package to test recovery from any of these failures. Because a zPDT Parallel Sysplex runs under z/VM, you can also easily simulate the failure of one or more devices. The great thing about running under z/VM is that any “device failure” can be created by running a VM command; there is no need to disconnect channels or power-off physical devices as is done when testing in a “real” Z environment.

Another audience is anyone who 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 queue manager is down, the messages can still be placed on (and retrieved from) the queue by any of the other queue managers in the queue sharing group. This feature provides a significant availability improvement, but it should be tested 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, the need for stringent change control is understandable. However, it *does* slow the evaluation of new functions and new products. In some cases, the process might be so onerous that implementation of many new functions might simply never happen.

Because zPDT is isolated from your real mainframe data (test and production), it should be easy to make a case to exclude your zPDT from change management. Also, because ADCD and zD&T come 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 Sysplex Extensions 2020 and previous versions

The previous zPDT sysplex support and support that is provided by the zPDT Sysplex Extensions 2020 includes the following differences:

- The November 2016 and May 2017 ADCD releases contained several modifications to make them more “sysplex-friendly.” Wherever suitable, this release of the Sysplex Extensions package uses these changes to reduce the installation effort.
- To reduce the number of volumes that you must add and allocate by running Linux commands, the package includes more DSS Dump-format z/OS volumes. It is also based on the ADCD z/VM 7.1 release, which uses fewer volumes than the previous ADCD z/VM releases.
- To reduce the time and complexity of modifying the SMS data and storage classes and storage groups manually, this release uses the NaviQuest function in DFSMS to perform those changes using batch jobs.
- This release of Sysplex Extensions 2020 includes VSAM RLS, together with a sample CICS workload that uses RLS to share VSAM files across the CICS regions in the sysplex.
- This release provides one CICS user volume and one IBM DB2 user volume. These volumes are 3390 model 9s, and provide more space than the previous two CICS and two DB2 volumes. Also, having only one CICS and one DB2 volume simplifies the installation effort. In accordance with the ADCD strategy to standardize on 3390 model 9 volumes, all but one of the Sysplex Extensions volumes that are provided with this package are model 9s.
- This release uses the z/OS Auto Reply function to automatically reply to messages that might delay the IPL process.
- Based on our experience with z/OSMF, this release runs the zFS function in the OMVS address space, rather than in its own separate address space. This does not remove any capabilities. In the zPDT environment, it results in significant CPU savings during the start of z/OSMF.
- This release tunes the SHUTS0Wn scripts to reduce the elapsed time to perform an orderly system shutdown.
- The SMFPRM member is set up to support both LOGSTREAM mode and DATASET mode. When running in Parallel Sysplex mode, SMF uses log streams by default. It also contains commented-out definitions for the SMF in-memory buffers used by the SMF Streaming support.
- 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 pre-installed policies.

- The previous version used the z/OS default WLM policy. This version uses the latest ADCD-supplied WLM policy as described at this web page (log in required). That policy includes changes to the WLM policy in support of the latest support in z/OSMF. It also ensures that z/OSMF performance is protected in a zPDT environment that uses zIIP processors.
- Dynamically allocated SVC dump data sets now contain the system name as part of the data set name.
- 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 terminal-owning regions (TORs), application-owning regions (AOR), IBM CICSPlex® SM address spaces (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 preferred practices.
- 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 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 Sysplex Extensions 2020" on page 35 provides detailed step-by-step instructions to help you install this package on a z/OS 2.4 base and manage the resulting sysplex environment.
- Chapter 5, "Sample Db2 data sharing environment" on page 99 provides step-by-step information to help you get the provided DB2 data sharing environment up and running.
- Chapter 6, "Sample CICSplex" on page 107 provides information to help you implement and use the provided CICSplex environment.

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

Important: The focus of the Sysplex Extensions is currently the base z/OS products, CICS, and DB2. No testing was done with IMS or IBM MQ, but we do not believe that these products encounter problems when running in a Sysplex Extensions environment.

2

System migration considerations

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

However, certain “good housekeeping” practices exist that can make it easier when you are ready to move to a new ADCD release or a new release of the Sysplex Extensions 2020. 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 scenario, you would download the volumes that are associated with a new ADCD release, perform an IPL with the new `sysres`, and everything would work as expected with no extra customization or effort. The other extreme scenario is that you must start from scratch when you install a new release and repeat the same customization process every time you move to a new release.

Although it is unlikely that you can achieve the ideal scenario, you can take actions to bring you closer to that scenario rather than the other, extreme scenario. 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 existing customization as possible, 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 and file systems. These data sets should be treated as if they are read-only, with the only updates being applied by SMP/E when you are applying service or performing an upgrade.
- System data sets that contain information that is specific to that system or sysplex, such as your IBM RACF® databases. These databases are provided by ADCD, but they also 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 and file systems, 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 (for example, the page data sets, RACF data sets, couple data sets (CDSs), Parmlib, Proclib, and others). Known as *system work volumes*, these volumes are supplied by ADCD. However, you often must perform some migration activity to carry forward some of the information in these data sets 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 minimize the number of volumes that are required for the complete ADCD environment. For example, the B4C551 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` volume is a good example of a *system volume*. Typically, you do not change any of the data sets on that volume. Taking the May 2020 ADCD release `sysres` volume as an example, the following data sets are exceptions to the recommendation about treating that volume as though it was read-only:

- `SYS1.PARMLIB`
- `SYS1.PROCLIB`

Although you *can* change the `SYS1.PARMLIB` and `SYS1.PROCLIB` data sets manually, we urge you to apply any Parmlib or Proclib changes to the `USER.Z24B.PARMLIB` and `USER.Z24B.PROCLIB` data sets. Those data sets are placed ahead of the `SYS1` equivalents in their respective concatenations. For example, if there is a PROGBO member in the `SYS1.PARMLIB` data set *and* in the `USER.Z24B.PARMLIB` data set, the member from the `USER.Z24B.PARMLIB` data set is used.

If you can adhere to this guidance, moving your SYSRES volume to the new ADCD level should be much easier and less time-consuming.

The product volumes (B4PRD1, B4PRD2, B4PRD3, and B4PRD4) are more examples of system volumes because the data sets that they contain should 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 (for example, B4SYS1, B4PAGA/B/C, B4USS1/2, and B4CFG1) are provided by ADCD, but they contain data that is specific to your environment. 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 (B4*nnnn*) reflect the May 2020 ADCD. If you install the Sysplex Extensions 2020 on top of a different ADCD release, you need to adjust for the different volsers used by your ADCD release. For example, the November 2017 ADCD release (the first release to deliver z/OS 2.3) used volsers such as A3*nnnn*.

Several of these volumes contain data that you might be willing to discard when you move to a new release, such as your JES2 spool, SMF data sets, page data sets, log recording data set (LOGREC) data sets, and dump analysis and elimination (DAE) data sets.

Tip: If you need to carry your RACF changes forward to a new ADCD release, you can use the RACFUNLD and DBSYNJCL jobs in the `ADCD.*.JCL` data set 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 in line with each other. See the DBSYNDOC member of `ADCD.*.JCL` for information about using the DBSYNC exec.

If you want to use a copy of the old RACF database, 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.

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[1] and SMS SCDS.

Tip: The ADCD-provided VATLST00 member mounts the B4SYS1 volume as a storage volume, meaning that things such as local TSO data sets are likely to be allocated on that volume. To save the work of moving those items to another volume when you replace B4SYS1 in the future, use a VATLSTxx member other than VATLST00 and ensure that it points at a local volume (WORK01, for example).

Review the contents of these volumes and identify how you can handle each data set when the time comes to upgrade to a new ADCD release.

The Sysplex Extensions 2020 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 Sysplex Extensions 2020 are contained on a pair of SMS-managed volumes (one volume for CICS and one 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.

Future releases of the Sysplex Extensions 2020 are expected to follow this model. That is, the volumes that contain the subsystem work data sets will be delivered with each new release. This configuration provides the option of replacing volumes with the new volumes (and cold starting 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.

zFS data sets

An increasing number of products provide their own zFS data sets. The zFS data sets display the same complexities that apply to many other software data sets:

- Some of them contain files that get updated on an ongoing basis - logs, for examples.
- Some of them are read only, and should be shared by all systems in the sysplex.
- Some of them require a separate instance for each system in the sysplex.
- Some of them require a separate instance corresponding to each set of target data sets for that product.

ADCD delivers over 70 zFS data sets, and it is outside the scope of this deliverable to address the sysplex considerations for all of those data sets. We *do* handle the sysplex needs of the zFS data sets that are delivered on the B4USSn volumes. For those file systems that require a separate instance for each system, we create a clone for the second instance and provide an updated BPXPRMxx member to reflect the new file names and mount points.

For the zFS files for which we do not provide clones or updated BPXPRMxx members, you can use the provided OMVSCLON and MKDIR jobs as a sample to create your own copies if you need to create such clones.

[1] There are several tools on the CBT tape that might help carry over catalog entries to a new Master catalog. In particular, consider the MCNVTCAT tool. For more information, see:
http://www.cbttape.org/cbtdowns.htm

User volumes

Most importantly, 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 application programs, your own data, load libraries, and JCL), place those data sets on volumes (called *user volumes*) that are not replaced by the new ADCD volumes or by the Sysplex Extensions. Also, ensure that they are cataloged in user catalogs that are also on the associated user volumes, and not on an ADCD-provided volume.

In as far as reasonable, place any new data sets that you create on a user volume that does 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

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.

If you IPL a new ADCD release with your existing 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 new 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:

- Identify the new data sets and add them to your current Master catalog.
- Identify 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, which 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 a “clean” Master catalog is to catalog all your user data sets in user catalogs, and place those catalogs on your user volumes.[2]

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

[2] To support RLS management of user catalogs, place any new user catalogs on SMS-managed volumes. Even if you do not intend to use this capability now, placing user catalogs on SMS volumes now can greatly simplify the migration to RLS-managed catalogs in the future and does not have any downside in the interim.

- LOGREC data sets
- Vitual input/output (VIO) journaling data set (often called `SYS1.STGINDEX`)
- Data sets in the Parmlib concatenation (for example, `SYS1.PARMLIB`, `ADCD.Z24B.PARMLIB`, and `USER.Z24B.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 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.Z24B.PARMLIB` member. Although this method works, the next ADCD release includes a new `ADCD.anna.PARMLIB` data set, which means that the change you made to the IGGCAT00 member in `ADCD.Z24B.PARMLIB` is lost.

Another option is to copy the entire IGGCAT00 member to your `USER.Z24B.PARMLIB` member. Because `USER.Z24B.PARMLIB` is ahead of `ADCD.Z24B.PARMLIB` in the Parmlib concatenation, the `USER.Z24B.PARMLIB` version is used and the copy in `ADCD.Z24B.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 when the IPL occurs 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.Z24B.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, 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. 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 valuable for those components that do support this use.

For more information about effectively managing 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

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 can significantly reduce 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. The highlighted text shows 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.S0W1.PLPA.PAGE,
      SYS1.S0W1.COMMON.PAGE,
      SYS1.S0W1.LOCALA.PAGE,
      SYS1.S0W1.LOCALB.PAGE,L),
PAK=00,                       SELECT IEAPAK00
PLEXCFG=ANY,
```

Example 2-2 shows an excerpt from the IEASYSB2 Parmlib member. The highlighted text in this example shows 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, with the exception that a system symbol is used in place of the system name. In this example, 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. This feature facilitates the use of one common member, rather than one per system.

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 on which system they are referenced.

In addition to Parmlib members, system symbols can be used in procedures, TSO logon procedures, 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 by using 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 Sysplex Extensions 2020 volumes, you can find examples of system symbols to simplify the members that are used for the sysplex.

2.1.5 Summary

The Sysplex Extensions provide a significant addition to the previously available ADCD sysplex support. In future releases, we plan to 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 or kynef@us.ibm.com.

3

Sysplex configurations

Important: This chapter is not intended as an introduction to z/OS sysplex concepts or operation. It is assumed 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 sysplexes that are available and the considerations for running a sysplex under zPDT.

This chapter includes the following topics:

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)
- XCF communication between 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.

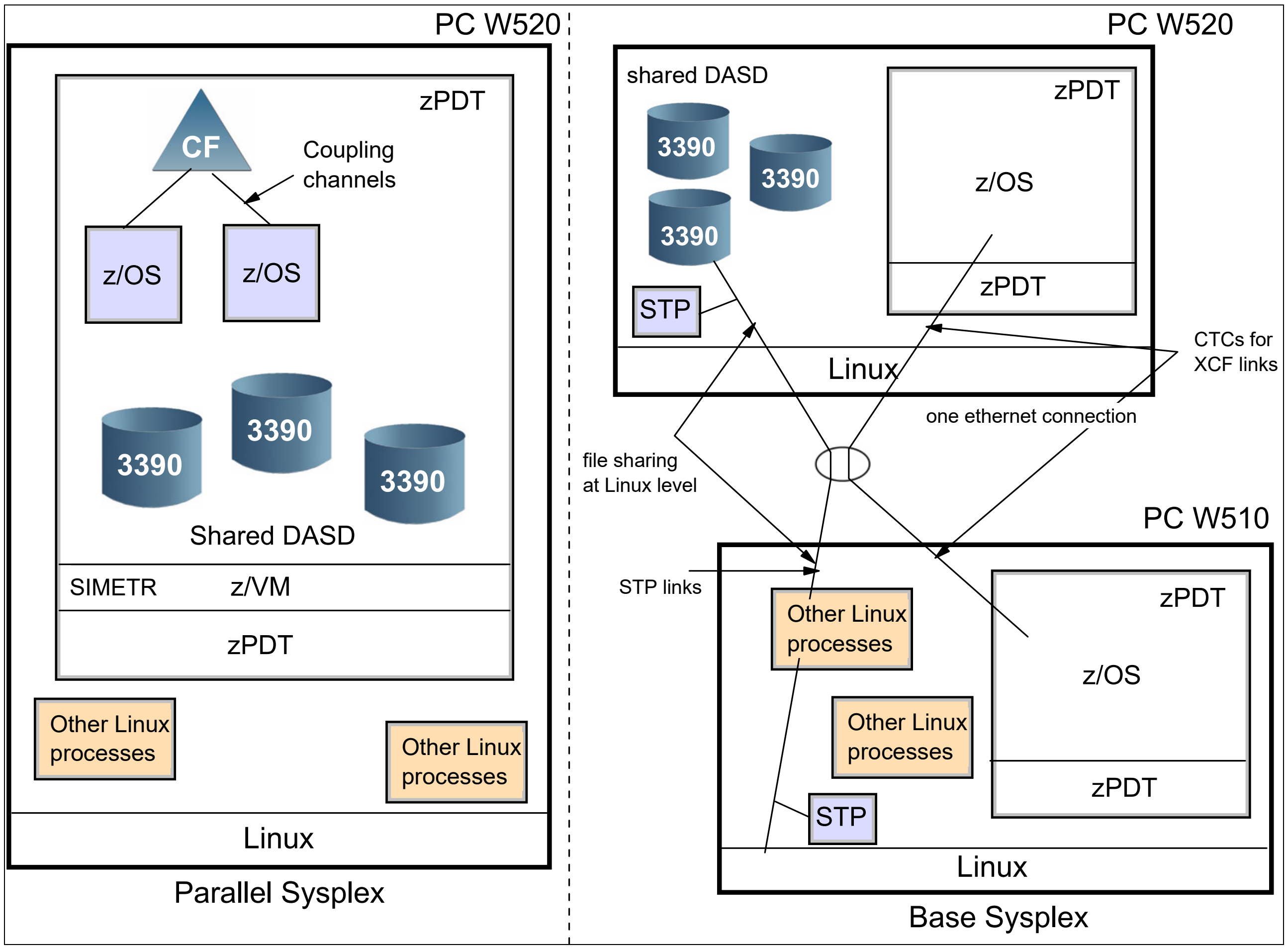


Figure 3-1 zPDT Parallel Sysplex and base sysplex

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[1] 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)[2] clock that is used by all z/OS guests. z/OS uses global resource serialization (GRS) (using the GRS Star lock structure in the CF) to coordinate data set sharing and other serialization across the members of the sysplex.

[1] This code is the same CFCC licensed code that is used on larger System z machines.
[2] A synchronized time-of-day clock is required for all members of a sysplex.

- The *base* sysplex configuration involves multiple Linux machines.[3] 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 through 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)[4] function (which was new in zPDT GA6).

  In Figure 3-1 on page 18, all of the emulated DASDs for all z/OS systems are 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 also 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 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 CF 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, which illustrates 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 IBM 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 other developers use the other system. This ability provides 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. In addition, configuring and operating a zPDT Parallel Sysplex requires basic z/VM administration skills. The advantages of sysplex operation must be weighed against the extra required skills.

[3] Multiple zPDT instances in the same Linux machine can also be used.
[4] 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. In addition, 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 using a sysplex 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[5] 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.[6] The same 1090 tokens are used by many IBM employees, with the same capabilities as the ISV users.
  - The 1091 tokens are used by zD&T clients. These tokens can have optional features enabled. The use of IBM Z CFs is an optional (priced) feature with 1091 tokens.
- Although the zPDT 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.[7]
- 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 IBM Z CPs that are allowed by one or more zPDT tokens that are on the PC, with an upper limit of eight CPs.[8]
- 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.

[5] It is more correct to refer to the *licenses* that are acquired through a token. However, for brevity it is referred to as a *token* here.
[6] Another license agreement covering z/VM usage might be required.
[7] This separation of license control is for zPDT releases dated 2Q13 or later.
[8] 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.
- The topic of z/OS system tuning is not addressed 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.

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, you must define at least one CF guest machine with a z/VM directory entry by using statements that are similar to the following example:

```
USER CFCC1 CFCC1 5000M 5000M 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 are 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 A9E 3390 DEVNO A9E 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 49.

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 IBM Z memory. IBM Z memory (in a zPDT environment) is virtual memory that is created by Linux. In principle, Linux virtual memory is only loosely related to the real memory of the PC. 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, typical zPDT documentation states that the defined IBM 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, 16 GB of real PC memory should be the minimum reasonable size, with more memory being better.[9]

With 16 GB of real memory, defining a 12 - 13 GB IBM 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.

[9] These numbers are minimum numbers. Much larger systems can 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, use *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) can 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. 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.

- IBM 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. Consider the following points:
  - If your zPDT system is running zIIP-eligible work *and* a physical core is used to back it, you might consider adding a “free zIIP” to your Devmap. For more information about the zPDT free zIIP support, see *IBM ZPDT Guide and Reference*, SG24-8205.

    Note: No benefit is gained by adding a zIIP if you do not have another core for it to use. For example, if you have a 4-core PC and already have three CPs defined to your zPDT system (leaving the fourth core for Linux use), adding a zIIP to your Devmap delivers little, if any, extra capacity.

  - The WLM policy that is included with ADCD (up to and including the May 2020 level) assigns z/OSMF and other system tasks to a Discretionary service class. Work in a discretionary service class *never* overflows to a general purpose CP. This makes sense when z/OS is running on a real z CPC. However, in a zPDT environment, ensure that no important work is assigned to a discretionary service class.
- z/VM overhead under zPDT. This overhead is greater than the z/VM overhead on a real IBM Z machine.

Allowing for these considerations, we found the performance of a Parallel Sysplex in our test environments 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 the emulated 3390 DASD were in one Linux PC. Our other Linux PC required all DASD access to go through the LAN. For read operations, active 3390 tracks might be contained in the local Linux caches.
- The sample configuration used NFSv4 for the shared file operation.[10] 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.
- Shared DASD performance that uses NFSv4 with 1 Gb Ethernet links and that placed all emulated DASD on one PC had performance concerns.

  We cannot predict whether the performance fits *your* requirements. However, you can try this configuration and see whether it is acceptable to you.

[10] NFS (as opposed to NFSv4) uses an older locking design. This package uses NFSv4 for base sysplex operation. NFS or NFSv4 might be used for the read-only DASD sharing.

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.

- 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 147.

NFS is not always a high-performance shared file system. It is attractive because it is widely available and simple to configure. The following methods can be used to address 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.
- 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, can 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 global resource serialization (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. The sample implementation does not explore this approach.

3.3 Sysplex goals

Chapter 4, "Installing the Sysplex Extensions 2020" 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 to 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. This environment is also designed 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.

The Sysplex Extensions 2020 are designed to adhere to IBM's sysplex preferred practices (or at least, those guidelines that are applicable to a zPDT environment). By providing a working example of such a configuration, 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

The 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 Sysplex Extensions 2020 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 Sysplex Extensions 2020" on page 35.
- The Sysplex Extensions 2020 package is based on the May 2020 ADCD z/OS 2.4 release. In principle, nothing is unique in this 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.
- This release of the zPDT Sysplex Extensions updates the sample CICSplex and DB2 data sharing environments. These environments are the most widespread sysplex environments in z/OS customers, so these environments can provide the most value to the largest number of users.

  The sysplex implementations include shared consoles, SMF log streams, and RRS log streams. They do not include IBM 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. The implementation also adds 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.

  Future releases of the zPDT Sysplex Extensions 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 you.
- For our Parallel Sysplex configuration, all DASD[11] is "owned" by z/VM and is shared by the z/OS guests. For the 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 2.4 IPL parameters can be used in this mode. The system name is `S0W1`.
  - The first system in a Parallel Sysplex. IPL parameter PS[12] is used, and the system name is `S0W1`.[13]
  - The second system in a Parallel Sysplex. IPL parameter PS is used, and the system name is `S0W2`.
  - The first system in a base sysplex. IPL parameter BS is used, and the system name is `S0W1`.
  - The second system in a base sysplex. IPL parameter BS is used, and 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.

[11] The emulated IBM Z DASD is referred to here. Other Linux files are in normal Linux locations.

[12] A zPDT IPL statement has the format `ipl a80 parm 0a82`*`xx`*; for example, where the two *`xx`* characters are the referenced "IPL parameter" in this description.

[13] The S0W1 name is set in the delivered ADCD z/OS system. The examples in this book do not alter this name.

- z/VM 7.1 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.
- The sysplex configurations are mostly determined by members in PARMLIB, PROCLIB, TCPPARMS, and VTAMLST. The ADCD system provides empty libraries, such as `USER.Z24B.PARMLIB` and `USER.Z24B.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 Sysplex Extensions 2020" on page 35.
- The Linux names (defined in the Linux system's `/etc/HOSTNAME` file) help identify the personal computer described in the examples in this book. The first base sysplex machine is named W520 (and it is also running the STP server), and the second base sysplex machine is named W510. The Linux names are not meaningful in a Parallel Sysplex environment unless they are needed for external IP name resolution.

Table 3-1 lists the intended IPL parameters and IP addresses.

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.S0W1
Base sysplex system 1	S0W1	BS or ST	192.168.1.80 10.1.1.2	ZFS.S0W1
Base sysplex system 2	S0W2	BS or ST	192.168.1.90 10.1.1.3[a]	ZFS.S0W2
Parallel Sysplex system 1	S0W1	PS	192.168.1.80 10.1.1.2	ZFS.S0W1
Parallel Sysplex system 2	S0W2	PS	192.168.1.90 10.1.1.3	ZFS.S0W2

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 IPL 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, there is a single IPL volume for z/OS. Different IPL parameters are used to start the different sysplex configurations that are provided (monoplex, base sysplex under z/VM, base sysplex spread over two PCs, and Parallel Sysplex).

The small numbers in Figure 3-2 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.

The CF0001, DB2001, and CICS01 volumes in Figure 3-2 are volumes that are provided as part of the zPDT Sysplex Extensions 2020. 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.

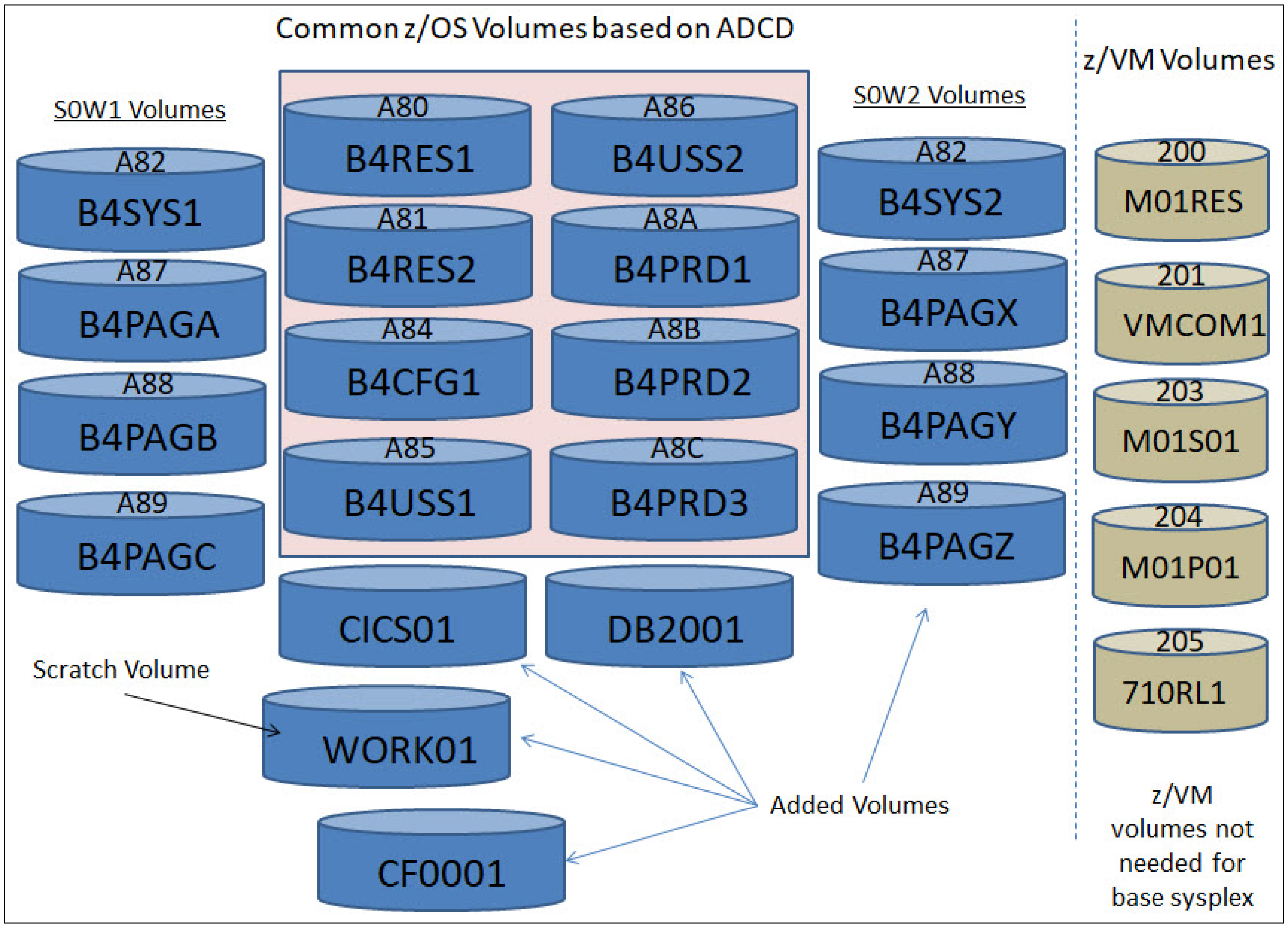


Figure 3-2 Total of 3390 volumes for sysplex

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 VSAM volume data set (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 z/OS cross-system coupling facility (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 two of 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, “Starting your Parallel Sysplex” on page 73.

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 steams 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
- *z/OS MVS Setting Up a Sysplex*, SA23-1399

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, take the following measures 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 used *always* for that log stream or is used *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, the archive log stream is defined and then the **`SETRRS ARCHIVELOGGING,DISABLE`** command is run 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). The base sysplex in this package 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.

This package sets 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 a Parallel Sysplex and a 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.

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 that 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[14], 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 log stream 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 log stream mode, update the appropriate `VTAMxx` Parmlib member to issue the **SETLOGRC LOGSTREAM** command during system initialization.

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.

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

The OPERLOG log stream 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 the `CNZ4201E OPERLOG HAS FAILED` message. Job LOGROPR, which is described in 4.4, “Starting your Parallel Sysplex” on page 73, allocates the OPERLOG log stream. If you want to enable the OPERLOG log stream after the log stream is defined, run the **`V OPERLOG,HARDCPY`** command.

If you want to stop using and delete its 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 the OPERLOG log stream 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 the OPERLOG log stream on that system.

For more information about the use of OPERLOG, see *z/OS MVS Planning: Operations*, SA23-1390.

3.4.4 System Management Facilities

The ability to write System Management Facilities (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 IBM 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.Z24B.PARMLIB` to specify `ACTIVE` rather than `NOACTIVE` and by 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 log stream mode, see the following resources:

- *SMF Logstream Mode: Optimizing the New Paradigm*, SG24-7919
- *z/OS MVS System Management Facility*, SA38-0667

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:

- `RVARY SWITCH`
- `RVARY ACTIVE`
- `RVARY INACTIVE`
- `RVARY DATASHARE`
- `RVARY 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.

4

Installing the Sysplex Extensions 2020

Important: This chapter is not intended as an introduction to z/OS sysplex concepts or operation. It assumes that readers understand the basic concepts and terminology that are 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 Sysplex Extensions 2020 package, which you can download from this web page.

For more information about downloading the additional material for this book, see Appendix E, “Additional material” on page 155.

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

This chapter includes the following topics:

[1] In theory, it should be possible to use the Sysplex Extensions 2020 with some small modifications to create a Parallel Sysplex that is based on an appropriately licensed zD&T environment.

4.1 Implementing a sysplex under zPDT

Important: This chapter is based on the May 2020 ADCD z/OS 2.4 system.

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 Sysplex Extensions 2020 package to create your multi-system Parallel Sysplex.

The following target configurations are supported:

- A Parallel Sysplex that is running under IBM z/VM
- A base sysplex that is running under z/VM
- A base sysplex that is running “native” under z/PDT, spread over two PCs

It is assumed that most people who 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 monoplex system into a two-way Parallel Sysplex. [2]

However, that path also includes most of the steps that are required for a base sysplex. Therefore, all users who 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, “Parallel Sysplex set up steps” on page 41.

If your objective is to have a base sysplex, complete the steps that are described in 4.5, “Other steps to create a base sysplex under z/VM” on page 78, or in 4.6, “Implementing a base sysplex without z/VM” on page 81, as appropriate.

4.2 Implementation overview

One of the objectives of this package is 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. The intent is also to make it easy for you to switch back to a standard ADCD environment if you need to.

Therefore, the package is as self-contained as possible and minimizes the number of changes you must make to your existing ADCD 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 monoplex environment.

Note: For more information about zPDT and ADCD, see *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

Review and become familiar with the contents of that book before you start the implementation of the Sysplex Extensions 2020. It contains a wealth of information that will prove valuable when customizing and fine-tuning the environment to your specific requirements.

[2] If you need more than two systems in your sysplex, you simply need to clone the installation jobs to do the corresponding setup for the third and subsequent systems.

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, “Parallel Sysplex set up steps” on page 41. Unless otherwise noted, all of the new PDS members that are described are created in `USER.Z24B.PARMLIB`, `USER.Z24B.PROCLIB`, and so forth.

Installation checklist: An installation checklist is provided in Table 4-4 on page 77. You can use the checklist to track your progress through the implementation steps. It is also helpful in ensuring that you do not miss any steps. You might want to print that table now, so that you can keep notes on it as you go along.

The following tasks that are performed to install the Sysplex Extensions 2020:

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 sysplex environment only. The implementation of the DB2 data sharing group is described in Chapter 5, “Sample Db2 data sharing environment” on page 99. The CICSplex environment implementation is described in Chapter 6, “Sample CICSplex” on page 107.

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 z/OS system under z/VM before moving to sysplex mode.
3. Download the volumes that are provided as part of the zPDT Sysplex Extensions 2020 package from this web page.

   Note: Starting with the 2020 release of the Sysplex Extensions, we intend to use the following device address ranges for the ADCD-delivered volumes and the Sysplex-Extensions-delivered volumes:

   - A80-A9F ADCD-delivered volumes.
   - AA0-AA7 Sysplex Extensions-delivered volumes.
   - AC0-ADF ADCD-delivered volumes *for previous ADCD releases*.
   - AE0-AEF Reserved for local volumes.

   You can override these ranges and use any address ranges that are defined in your zPDT Devmap and z/OS IODF.

   Table 4-1 lists the volume serial numbers and virtual address that are used for them, and provides a brief description of the contents of each volume.

Table 4-1 Volumes that are delivered by the zPDT Sysplex Extensions 2020 package

Volser	Device number	Use
B4PAGX B4PAGY B4PAGZ	AA0 AA1 AA2	Local, COMMON, and PLPA page data sets for the second z/OS system.
B4SYS2	AA3	System-specific data sets for the S0W2 system.

Volser	Device number	Use
CF0001	AA4	Couple data sets (CDSs) for base and Parallel Sysplex. PDSs containing sample JCL, Parmlib, Proclib, and other required data sets.
CICS01	AA5	SMS-managed volume containing data sets required by the CICSplex subsystems that are provided by this package.
DB2001	AA6	SMS-managed volume containing the data sets required by the DB2 data sharing group that is provided by this package.
WORK01	AA7	Work volume.

Note: Appendix A, "Sample definitions" on page 115 contains a complete list of the volumes that are used in the example sysplex and the device number used for each one. Additionally, Table A-1 on page 126 lists all the device numbers and device types defined in the ADCD-provided IODF.

The B4PAGX/Y/Z, B4SYS2, and WORK01 volumes are provided as empty volumes. In later steps, the relevant data sets are allocated on those volumes.

Tip: If you experience PGT004 wait states in z/VM, the cause it might be that more z/VM paging volumes are needed. For more information about adding the paging volumes to z/VM, see IBM Knowledge Center.

In our testing, we encountered HCPPGT400I and HCPPGT401I 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 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, "Establishing a known restart point" on page 44.
5. After you download and extract the Sysplex Extensions 2020 volumes, update your zPDT device map (`devmap`) file to add the new DASD volumes.

   You might want to add commands to your `devmap` file to start more x3270 sessions automatically. Because you are running z/VM and another z/OS instance, it is likely that you need more 3270 sessions than exist now.

   You also must define the Open Systems Adapter (OSA) device numbers that are to be used to communicate between the z/OS guests and Linux, and between the z/OS guests and the rest of your network. For more information, see Appendix A, "Sample definitions" on page 115.
6. The Parallel Sysplex environment that is provided with this package uses two z/OS virtual machines (VMs), called `SOW1` and `SOW2`, and two Coupling Facility VMs, called `CFCC1` and `CFCC2`. All of these VMs are included in the `VM` directory that is provided with the ADCD z/VM package.

A small change to the VM directory might be required to identify the volumes that contain the CDSs. Also, the S0W1 and S0W2 users are defined with 5000 MB of virtual memory, which should be increased. The precise amount depends on how much work you plan to run concurrently in the sysplex, and the amount of real memory available in the PC that zPDT is running in. We used 9000 MB each for our testing.

7. 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), **`IMPORT CONNECT`** the new user catalogs and create ALIASes for the HLQs in those catalogs.[3]
8. Create the system-specific system data sets for system S0W2, such as paging, virtual I/O (VIO), System Management Facilities (SMF), and LOGREC.
9. Clone system-specific zFS file systems for the second (S0W2) system.
10. Create trace and output message data sets for zFS.
11. 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, allocate the `S0W2` copy of that data set to avoid a JCL ERROR when the Health Checker is started on `S0W2`.
12. The Sysplex Extensions 2020 package provides a new set of CDSs to be used 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, you re-catalog the Sysplex Extensions 2020-provided CDSs in your Master catalog. Also, the VSAM RLS Share Control Data Sets are re-cataloged in this step.
13. The ADCD-provided SMS configuration must be updated to add the following items:
    - The name of the second system and the sysplex name.
    - A cache set and a cache structure for VSAM RLS.
    - A lock set and second lock structure for VSAM RLS.
    - Two data classes for system logger (LOGR) data sets, one for DB2 data sets, and one for zEDC-compressed data sets.
    - A new storage class for CICS and DB2 runtime data sets (logs, journals, and so on) and a new storage class for the VSAM data sets that will be accessed in RLS mode.
    - New storage groups for the CICS and DB2 volumes.

    In addition to the changes to the SMS configuration, you also must update the ACS routines to assign the appropriate SMS objects to each data set.

    In this step, we use NaviQuest in a batch job to make as many of the changes as possible.
14. The JES2 parameters must be altered to define a Multi-Access Spool (MAS) system with two members (`S0W1` and `S0W2`).

    Tip: In this step, you update the JES2PARM member that is used by your *driving system*. Therefore, you must be especially careful not to create any syntax errors that would stop your current system from starting.

[3] These installation instructions assume that you will use your existing Master catalog for both monoplex and multi-system sysplex environments. The Sysplex Extensions do not provide a new Master catalog.

15. To provide flexibility to switch back and forth between sysplex mode and monoplex mode, an attempt was made to isolate all Parmlib changes to a new set of members in `USER.Z24B.PARMLIB`, rather than changing any of the members in the `ADCD.xxxx.PARMLIB` or `SYS1.PARMLIB` data sets.

    In this step, you copy the Sysplex Extensions 2020-provided Parmlib members to `USER.Z24B.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, this package uses the following suffixes:

BS	Base sysplex under z/VM
PS	Parallel Sysplex under z/VM
ST	Base sysplex spread across two or more PCs

16. Create PROCLIB members for the SHUTSYS, VTAMPS, TCP/IP, and TFSPROC started tasks.
17. Copy and customize TCPPARMS members, as described in 4.3.20, “Creating TCPPARMS” on page 71.
18. Create VTAMLST members for ATCCONPS, ATCSTRPS, and OSATRLx.
19. Perform an IPL for the first member of your new sysplex (`S0W1`).
20. Submit jobs to define the log streams for LOGREC, OPERLOG, Health Checker, Resource Recovery Services (RRS), and SMF.
21. Create a mount point in the UNIX file system for the system-unique files for system S0W2.
22. IPL the second member of your new sysplex.

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 channel-to-channel (CTC) for inter-system cross-system coupling facility (XCF) communication. For ease of operation, define the CTCs in the VM directory.
2. Another aspect that is related to the lack of a CF is Global Resource Serialization (GRS). Because there is no CF, GRS must run in Ring mode rather than Star mode. Update the IEASYSxx member 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 common 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 RRS 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 Parallel Sysplex set up steps

This section provides describes the steps to use the Sysplex Extensions 2020 to create a Parallel Sysplex or a 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 to create a base sysplex under z/VM” on page 78 and should be used *only* if your target environment is a base sysplex.

4.3.1 Installing the base ADCD system

The information presented here assumes that you are familiar with ADCD, and likely already have an ADCD-based z/OS system running in monoplex mode. If you do not, create one by using the standard ADCD documentation.

Note: All the documentation and samples provided in this edition of SG24-8386 are based on the May 2020 version of the ADCD z/OS 2.4 deliverable. Ensure that you are using that version before proceeding with the steps described in this book.

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

To answer zPDT questions that might arise during the installation, use *IBM zPDT Guide and Reference System z Personal Development Tool*, SG24-8205.

Available resources: If you encounter problems and cannot resolve them using the information in the resources listed here, contact the zPDT forum for help in determining the nature of the issue.

The forum is a simple discussion group that is supported on a best efforts basis. It is intended for discussions about zPDT itself and ADCD questions. It is not intended as a vehicle for more complex “how to” issues related to System z operating systems and software products.

This package uses the ADCD May 2020 z/OS 2.4 release as the base. We downloaded the following subset of the ADCD-provided volumes:

- B4RES1
- B4RES2
- B4SYS1
- B4USR1
- B4CFG1
- B4USS1
- B4USS2
- B4PAGA
- B4PAGB
- B4PAGC
- B4PRD1
- B4PRD2
- B4PRD3
- B4PRD4
- B4C551

- B4DBAR
- B4DBC1
- B4DBC2
- B4ZWE1

These volumes provide z/OS and the common products, such as compilers.[4] We also suggest that you download 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 complete the IPL process.

Also, if you expect to need to install service on z/OS, restore the three DLIB volumes (B4DIS1 B4DIS2, and B4DIS3). We installed the volumes that are used by CICS V5.5 (B4C551) and DB2 V12 (B4DBAR, B4DBC1 and B4DBC2). However, these volumes 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.

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), **install the ADCD z/VM package now**.

For our Parallel Sysplex, we used the ADCD z/VM 7.1 release.[5] We downloaded the following volumes:

- 710RL1
- M01RES
- M01P01
- M01S01
- VMCOM1

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 suggest that you also use these device numbers):

```
device 0200 3390 3990 M01RES
device 0201 3390 3990 VMCOM1
* device 0202 no longer used by z/VM 7.1 ADCD
device 0203 3390 3990 M01S01
device 0204 3390 3990 M01P01
device 0205 3390 3990 710RL1
```

[4] The B4PRDn 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.

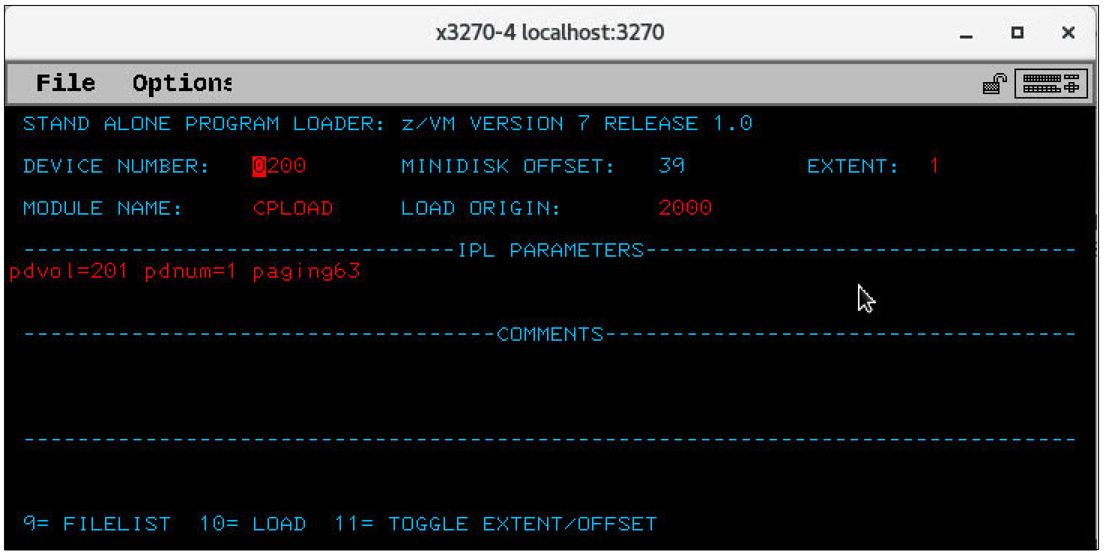
[5] If you are using zPDT release 7, you must use z/VM 6.3, or z/VM 6.2 with the APARs that are required to support IBM z13®. These APARs are required even though you are not running on a real z13. If you are using zPDT release 8 or 9, you must use z/VM 6.4 or later with the APARs to support z14.

For more information about the use and maintenance of the z/VM system, see the z/VM product documentation.

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.

Complete the following steps to load z/VM under zPDT. You can start the 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:

1. Run the zPDT **`LOADPARM 0700`** command to define 3270 device 700 as the VM IPL console.
2. Run the zPDT **`IPL 0200`** command to IPL z/VM. 200 is the device number of the M01RES volume.
3. On the x3270 session that corresponds to device 700, specify the device number of the VMCOM1 volume on the PDVOL parameter on the z/VM IPL screen, as shown in Figure 4-1. The sample devmap statements that are provided with this package assign device number 201 to the VMCOM1 volume. Then, press **`PF10`** to proceed with loading z/VM.



```
STAND ALONE PROGRAM LOADER: z/VM VERSION 7 RELEASE 1.0

DEVICE NUMBER:    0200       MINIDISK OFFSET:   39          EXTENT:  1

MODULE NAME:      CPLOAD     LOAD ORIGIN:       2000

--------------------------------IPL PARAMETERS--------------------------------
pdvol=201 pdnum=1 paging63


-----------------------------------COMMENTS-----------------------------------



------------------------------------------------------------------------------


9= FILELIST  10= LOAD  11= TOGGLE EXTENT/OFFSET
```

Figure 4-1 z/VM IPL Panel

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 VMs.
- Autologging the CF VMs.
- Updating the VM directory and by running 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, install z/VM and run your ADCD z/OS system in monoplex mode under z/VM. This approach gives you an opportunity to get familiar with 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.[6]

4.3.3 Downloading and creating 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 Sysplex Extensions 2020.

The zPDT Sysplex Extensions 2020 uses the following volumes:

CF0001	This volume contains CDSs, sample JCL, and sample definitions.
CICS01	This volume contains the user catalog and data sets for the CICSplex regions as well as the data sets for the PSTE sample CICS application.
DB2001	This volume contains the data sets for the DB2 data sharing subsystems.
B4PAGX	This starts as an empty volume. It includes page data sets for the second z/OS system (S0W2).
B4PAGY	This empty volume contains page data sets for the second z/OS system.
B4PAGZ	This empty volume contains page data sets for the second z/OS system.
B4SYS2	This empty volume contains system and work data sets that are used by the second z/OS system.
WORK01	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. To maintain consistency with the ADCD strategy of using 3390-9s for all the volumes they provide, all the remaining volumes that are provided by the package are 3390-9s.

All volumes are provided in a single .gz file called `syspext_8386-02_tar.gz`, which is created by running the Linux **`tar`** command. The volumes can be downloaded from this website. Download the .gz file into the directory that contains your ADCD z/OS volumes. The .gz file requires approximately 150 MB of disk space, and the extracted files require 57 GB of disk space.

After the download completes, run the following command to extract the Sysplex Extension volume files:

```
tar -xzvf syspext_8386-02_tar.gz
```

You now have the eight volumes that are provided by this package.

4.3.4 Establishing a known restart point

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

[6] All of the z/OS parts of the Sysplex Extensions 2020 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 performing the z/VM-related steps separately. However, if your target environment is a base sysplex that spans multiple PCs, it is not necessary to use z/VM, so you can skip the z/VM-related steps.

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

An alternative method 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.

To help you get started with the disk versioning function, a basic Linux script is included to enable versioning support for the z/OS and z/VM disks. The script is shown in “zPDT Disk versioning sample script” on page 131. You can customize this script to include more volumes (your own z/OS volumes, for example), or you can create a corresponding script to accept the outstanding changes and create a new 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, at a minimum you must create a backup of the following data sets:

- `ADCD.Z24B.PARMLIB` (or the corresponding data set for your level of ADCD)
- `USER.Z24B.PARMLIB`
- `USER.Z24B.PROCLIB`
- `USER.Z24B.TCPPARMS`
- `USER.Z24B.VTAMLST`

System restart: Your z/VM and z/OS system likely is down at this point. Therefore, make a note to backup those data sets after you restart 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 to 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 when you do not have one. Therefore, taking a few minutes to create a backup now is a good investment of your time. To make it even easier, you can use the JCL that is shown in Example 4-1. The only required changes to this example should be to update the SET HLQ statement to specify the HLQ that you want to use for those data sets.

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

```
//BACKUP JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*
// SET   HLQ=userid
//*
//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.Z24B.PARMLIB
//OT1      DD DISP=(NEW,CATLG),DSN=&HLQ..Z24B.PARMLIB.BKUP,
//            UNIT=SYSDA,
//            LIKE=ADCD.Z24B.PARMLIB
//IN2      DD DISP=SHR,DSN=USER.Z24B.PARMLIB
//OT2      DD DISP=(NEW,CATLG),DSN=&HLQ..USER.Z24B.PARMLIB.BKUP,
//           UNIT=SYSDA,
//           LIKE=USER.Z24B.PARMLIB
//IN3      DD DISP=SHR,DSN=USER.Z24B.PROCLIB
//OT3      DD DISP=(NEW,CATLG),DSN=&HLQ..USER.Z24B.PROCLIB.BKUP,
//           UNIT=SYSDA,
//           LIKE=USER.Z24B.PROCLIB
//IN4      DD DISP=SHR,DSN=USER.Z24B.TCPPARMS
//OT4      DD DISP=(NEW,CATLG),DSN=&HLQ..USER.Z24B.TCPPARMS.BKUP,
//           UNIT=SYSDA,
//           LIKE=USER.Z24B.TCPPARMS
//IN5      DD DISP=SHR,DSN=USER.Z24B.VTAMLST
//OT5      DD DISP=(NEW,CATLG),DSN=&HLQ..USER.Z24B.VTAMLST.BKUP,
//           UNIT=SYSDA,
//           LIKE=USER.Z24B.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
/*
```

Ensure that the job ends with return code 0 before you make changes to these data sets.

Starting over

If you find yourself in a position where you want to return to the beginning and start the installation of the Sysplex Extensions 2020 over again, delete (or rename) *all* of the following volumes:

- CF0001
- CICS01
- DB2001
- B4PAGX
- B4PAGY
- B4PAGZ
- B4SYS2
- WORK01

Then, run the **`tar -xvzf`** command against the downloaded .gz file again to recreate those volumes.

4.3.5 Updating the devmap file 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 virtual 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 input/output definition file (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 126.)

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

```
device 0aa0 3390 3990 B4PAGX
device 0aa1 3390 3990 B4PAGY
device 0aa2 3390 3990 B4PAGZ
device 0aa3 3390 3990 B4SYS2
device 0aa4 3390 3990 CF0001
device 0aa5 3390 3990 CICS01
device 0aa6 3390 3990 DB2001
device 0aa7 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/B4RES1), you must adjust the device statements accordingly.

When you are updating the devmap, take a note of the device number that you assign to the new volumes. You can enter the file information in Table 4-2. The devmap file points only to a Linux file. The file name does not necessarily *have* to match the volser of the volume contained in the file. However, it is *strongly* recommend that the name of the Linux file matches the volser of the z/OS or z/VM volume that is in that file. For example, the Linux file B4SYS2 should contain the z/OS volume that is called B4SYS2. Failing to use this naming convention can lead to confusion and much wasted time later on.

Table 4-2 zPDT Sysplex Extensions 2020 volumes

z/OS Volser	Linux file name	Device number	Notes
B4PAGX			
B4PAGY			
B4PAGZ			
B4SYS2			
CF0001			
CICS01			
DB2001			
WORK01			

While you are updating the devmap file to add the new volumes, take the opportunity to add commands to the devmap file to start some other x3270 sessions automatically. When you are running z/OS under z/VM, you are likely to 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 VMs

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

```
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 must update the `devmap` file to add the definitions of the Open Systems Adapter (OSA) devices to connect your systems to the network. For more information about using the statements, see 4.3.17, "Network definitions" on page 67. For now, add the following definitions:

```
[manager]
# Add definition for IP tunnel connection between Linux and zPDT
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]
# Use find_io command to find path number of your network connection
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 provided VTAMLST and TCPPARMS members. Unless you are experienced with VTAM and TCP, 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 4.3.17, "Network definitions" on page 67.

If your z/OS system is running, 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` file information. Devices that are not defined in the `devmap` file at the time the **`awsstart`** command is issued cannot be added dynamically. Therefore, you must stop and restart zPDT to make them known to zPDT.

When the new `devmap` file is loaded successfully using the **`awsstart`** command, the next step is to complete an IPL for 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 z/VM completes the IPL, 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. The volumes that were delivered by the zPDT Sysplex Extensions 2020 should be online. The output from the **`Q`** command displays something that is similar to `DASD 0AA4CF0001`, where `0AA4` is the device number, and `CF0001` is the volser of the volume that is mounted on that device. Depending on the level of the z/VM system that you are using, you might also have to issue the following commands (but there is no harm in running these commands in any case):

`FORCE TCPIP`	This command is required so that the network interfaces are available to the z/OS guests.
`DET A80-AFF SYSTEM`	The latest z/VM ADCD system attaches the z/OS device numbers to the system. They must be detached from the system so that the z/OS guests can use those devices for minidisks.

4.3.6 Configuring z/VM guests

The Parallel Sysplex requires four VMs (also known as *guests*):

- Two coupling facility VMs, called *CFCC1* and *CFCC2*
- Two z/OS VMs, called *S0W1* and *S0W2*

If you want to run the ADCD system as a monoplex under VM, use a VM called `BASEAD`. The z/VM 6.4 and 7.1 ADCD packages include the definitions for all these VMs in the supplied VM directory. The source for the ADCD-supplied directory is stored in the `USER DIRECT C` file on the MAINT710[7] user ID. The initial password for these user IDs is ZVM710, which should be changed as soon as you start using these 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 full-volume minidisks belonging to a VM 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 118.

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.

Note: *If* you place the CF0001 volume on a device number *other* than AA4, you must make a 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 channel programs for that device. This configuration ensures that any working allegiance is not severed by z/VM ending the channel program “early.” Any program that performs atomic I/Os needs working allegiance. In general, it should be allowed to default 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 sysplex CDSs.

[7] Note that this is a change. Previous levels of the z/VM ADCD used MAINT as the system programmer userid, rather than MAINT710.

If you plan to use the System-Managed CF structure duplexing function, including the new Asynchronous System-Managed Duplexing capability, you must have links between the two coupling facility VMs. Check that the entries for the CFCC1 and CFCC2 user IDs contain the following line immediately after the CONSOLE statements:

```
SPECIAL MSGP 1600 targid
```

Where `targid` is the name of the CF to which you want to connect. For example, the CFCC1 entry should contain `SPECIAL 1600 MSGP CFCC2`. Use the same address (1600) for both CF VMs.

After you make any required changes to the directory entries, close the file and 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 VMs.

4.3.7 Completing the IPL for the z/OS driver system

The next step is to start the z/OS system that will be used to perform most of the remaining customization steps.

Log on to the BASEAD user ID on z/VM (make sure that the `S0W1` system is not up when you logon to BASEAD). When you do, you see many `DASD 0Axx offline` messages. Because those messages likely relate to the spare devices as described in 4.3.6, “Configuring z/VM guests” on page 49, they can be ignored.

Now, complete the IPL for z/OS by running the following commands:

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

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

During the IPL, you might encounter message ILR031A. This message is produced when the current system name is different to the name of the last system to use the page data sets. Because you are bringing the system up under the BASEAD user ID, you must reply **`R 00,CONTINUE.`**

You also are likely presented with message IXC420D, which prompts you to decide if you want to initialize the sysplex. Assuming that this system is the only system that uses the current set of sysplex couple data sets, reply **`R 00,I`**.

The next step is to log on to TSO using the IBMUSER or ADCDMST user IDs. To get to the z/OS TSO logon window, run the **`DIAL BASEAD`** command in the COMMAND area of any of the available VM windows.

4.3.8 Importing user catalogs

Most of the data sets that are provided by the zPDT Sysplex Extensions 2020 are cataloged in user catalogs on the Sysplex Extensions 2020-provided volumes. To make those user catalogs and the data sets they reference available to your system, run an `IDCAMS IMPORT CONNECT` to make them known to your existing Master catalog.

Note: The Sysplex Extensions 2020 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 additional benefit.

Job IMPCONN in data set `SYSPLEX.PARALLEL.CNTL` on the CF0001 volume performs the IMPORT CONNECTs for all user catalogs provided with this package and defines the related aliases for you.

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

4.3.9 Creating system-specific system data sets

Several system data sets cannot be shared between systems. In this step, you allocate those data sets for the S0W2 system. Figure 4-2 shows the contents of the B4PAGA/B/C/X/Y/Z and the B4SYS1 and B4SYS2 volumes at the start of this step.

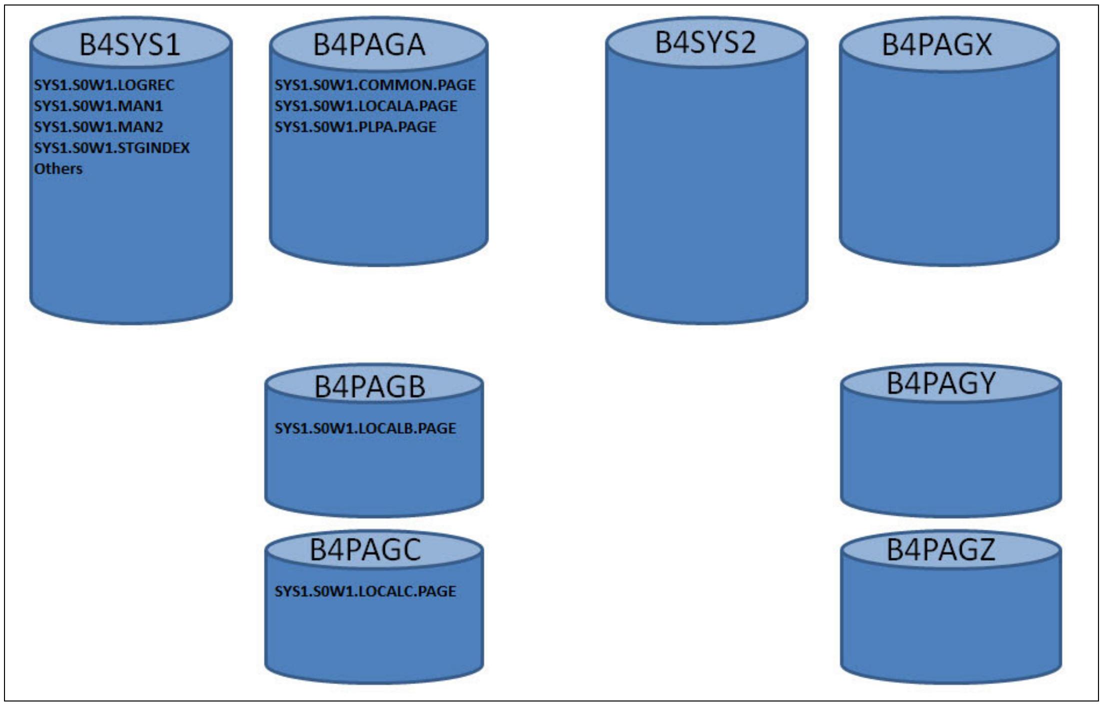


Figure 4-2 System volumes and system data sets

Figure 4-3 shows the contents of those volumes *after* you complete this step. You see that there are data sets for the S0W1 and S0W2 systems when running in sysplex mode. The same S0W1 data sets can also be used when running in monoplex mode.

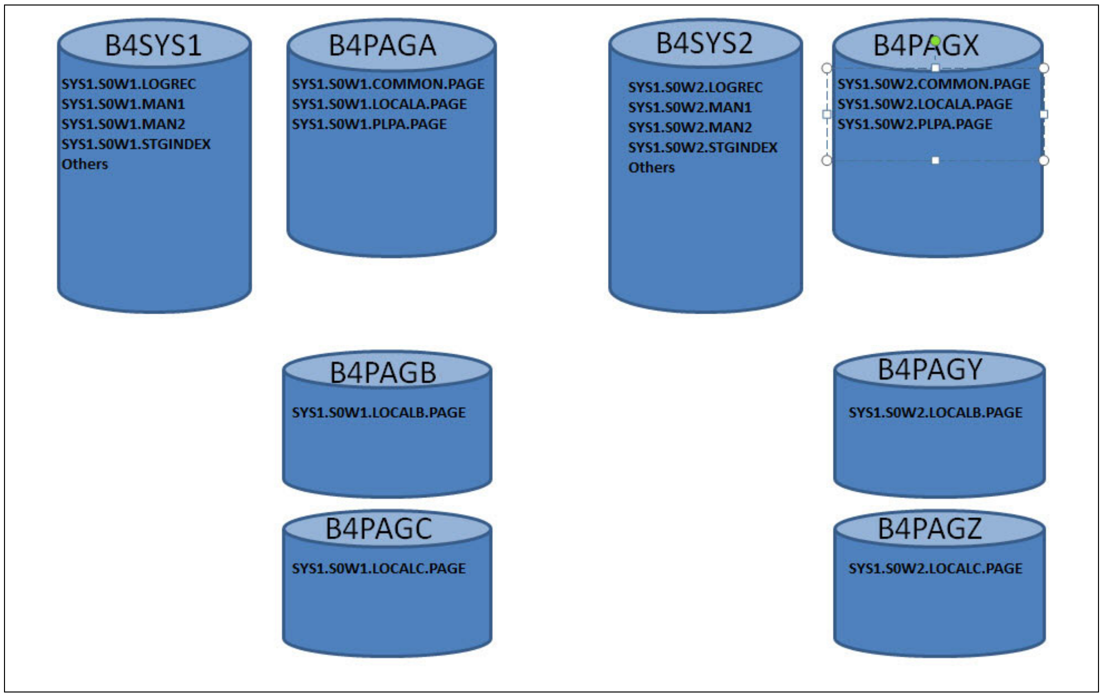


Figure 4-3 System volumes and system data sets

Data set `SYSPLEX.PARALLEL.CNTL` (you can access it by using the normal catalog search order after you run the IMPCONN job) contains a job called DEFPAGE to define the Paging, VIO, SMF, and LOGREC data sets for system S0W2. The page data sets are allocated on the B4PAGX/Y/Z volumes. The other data sets are allocated on the B4SYS2 volume. All of these data sets are cataloged in the Master catalog.

Submit the DEFPAGE 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. Also, you might see various messages related to I/O delays while the new page data sets are being formatted. These messages can include Missing Interrupts, “waiting to access JES2 chkpt”, and possibly similar messages on the z/VM console. These messages are caused by the high level of disk write activity caused by the DEFPAGE job. You can ignore these messages, and they should stop when the DEFPAGE job completes.

The IEASYSxx and IEASYMxx Parmlib members that are provided with this package support any number of systems if you use the same naming convention for your system-specific data sets that are used by the DEFPAGE job.

4.3.10 Creating zFS file systems for system S0W2

A subset of the zFS data sets is system-specific. The Sysplex Extensions 2020 package provides a job called OMVSCLON in `SYSPLEX.PARALLEL.CNTL` to create a copy of these data sets for system `S0W2` and place them on volume `B4SYS2`.

The names of many of the zFS data sets provided with the May 2017 ADCD release were changed to make them much better suited to a sysplex environment, while also being ideal for a single-system environment. This new naming convention allows you to have a single BPXPRMxx member that uses system symbols to specify the correct files for each system. Also, the file names indicate how the file is being used by looking at the data set name.

The OMVSCLON job allocates and formats a system root data set for the `S0W2` system. It then uses DFSMSdss to create a second copy of the system-specific zFS files. Submit the OMVSCLON job now. The corresponding BPXPRMxx members were updated to reflect the new data set names and are copied to the USER.Z24B.PARMLIB data set later.

4.3.11 Creating data sets for zFS started tasks

The zFS started tasks use two data sets each to record information about their activity (messages and trace activity). Submit the ZFSLOGS job in `SYSPLEX.PARALLEL.CNTL` now to create those data sets.

4.3.12 Creating Health Checker persistent data file for S0W2

The z/OS Health Checker persistent data file (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 that you might otherwise be unaware of.

The ADCD system includes a data set called `ADCD.S0W1.HZSPDATA` that is used by the Health Checker started task on system S0W1. This step allocates 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.

4.3.13 Recataloging Master catalog data sets

A few data sets (the new Sysplex Extensions 2020-supplied CDSs and the VSAM RLS SHCDS data sets) are provided on the Sysplex Extensions 2020-supplied CF0001 volume and 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 sysplex CDS data sets have a high-level qualifier of PARALLEL. Ensure that an alias of PARALLEL is not already defined in your Master catalog. Then, submit the RECATLG job and ensure that it completes with return code 0.

4.3.14 Making the necessary SMS changes

There are 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. The VSAM RLS data sets also must be SMS-managed. In addition, for ease of management and to make them self-contained, the entire CICSplex environment is packaged on an SMS-managed volume (CICS01).

As a result, multiple changes to the standard ADCD SMS configuration are needed. You already have an SMS SCDS and ACDS (they are provided by ADCD) and might have made changes to them. For that reason, replacement SMS control data sets are not included because their use would regress your changes. Also, there is no mechanism to merge two sets of SMS control data sets.

However, NaviQuest was used to update the SMS configuration using batch jobs. The jobs are packaged so that each job submits the subsequent job, meaning that you need to submit only the first job in the sequence.

The following jobs are provided in the `SYSPLEX.PARALLEL.CNTL` data set:

SMS0	Allocates an ISPF table data set that is used by all the subsequent jobs.
SMS01	Changes the share options of the ADCD-supplied SMS control data sets to 3,3 to support cross-system sharing.
SMS1	Adds the second z/OS system (S0W2) and the sysplex name (ADCDPL) to the SMS configuration.
SMS1011	Adds a lock set and associated lock structure for use by VSAM RLS.
SMS1012	Adds a cache set and associated cache structures for use by VSAM RLS.
SMS201	Adds a storage group called CICFILES for use by CICS.
SMS2011	Adds the CICS01 volume to the CICFILES storage group.
SMS202	Adds a storage group called DB2FILES for use by DB2.
SMS2021	Adds the DB2001 volume to the DB2FILES storage group.
SMS3011	Adds a data class for the Logger staging data sets (which must have a 4 K CI Size).
SMS3012	Adds a data class for Logger offload data sets (which have a recommended CI Size of 24 K).
SMS3013	Adds a data class for extended format data sets for use by DB2.
SMS4011	Adds a storage class (PSADDON) for any data sets that must be SMS-managed.
SMS4012	Adds a storage class for VSAM RLS data sets.

Submit the SMS0 job now and ensure that all the subsequent jobs are submitted and complete successfully.

Note: For more information about NaviQuest, see Chapter 22 of *z/OS DFSMS Storage Administration*, SC23-6860. You will find sample NaviQuest jobs in the `SYS1.SACBCNTL` data set, and the NaviQuest REXX executables are available in the `SYS1.DGTCLIB` data set.

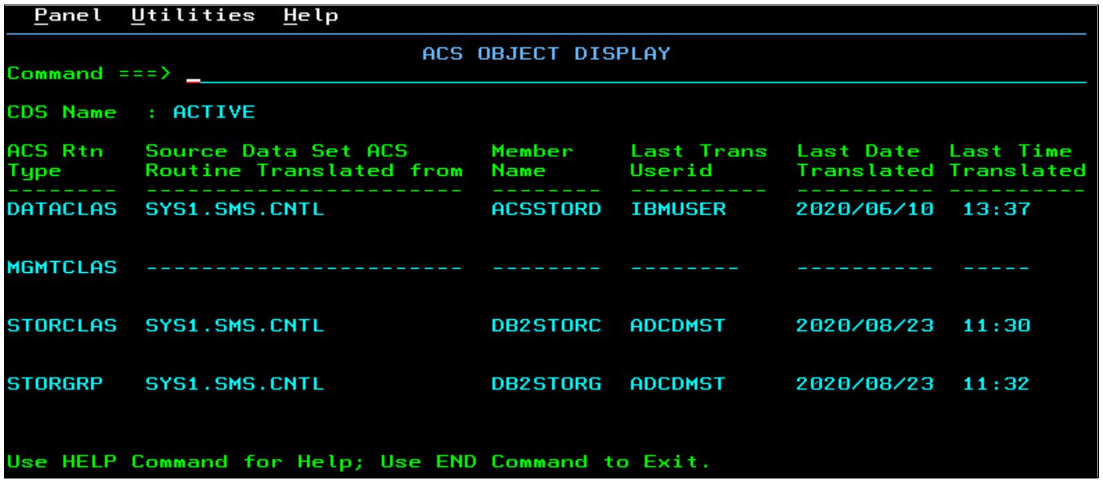
Updating ACS routines

The SMS data class, storage class, and storage group ACS routines must be updated to reflect the new storage classes and storage groups.

The ADCD-provided SMS ACS routines are kept in data set `SYS1.SMS.CNTL`. The following members contain the source for the active ACS routines:

Data class	ACSSTORD
Storage class	DB2STORC
Storage group	DB2STORG

You can 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 are presented with a list of routines and their location, as shown in Figure 4-4.



```
  Panel  Utilities  Help
                                ACS OBJECT DISPLAY
Command ===> _

CDS Name   : ACTIVE

ACS Rtn    Source Data Set ACS        Member     Last Trans   Last Date   Last Time
Type       Routine Translated from    Name       Userid       Translated  Translated
--------   -----------------------    --------   ----------   ----------  ----------
DATACLAS   SYS1.SMS.CNTL              ACSSTORD   IBMUSER      2020/06/10   13:37

MGMTCLAS   -----------------------    --------   --------     ----------   -----

STORCLAS   SYS1.SMS.CNTL              DB2STORC   ADCDMST      2020/08/23   11:30

STORGRP    SYS1.SMS.CNTL              DB2STORG   ADCDMST      2020/08/23   11:32


Use HELP Command for Help; Use END Command to Exit.
```

Figure 4-4 ISMF Display of ACS routines information

Note: If you have *not* modified the ADCD-provided ACS routines, submit the COPYACS job in `SYSPLEX.PARALLEL.CNTL` and skip to "Activating all your SMS changes" on page 58. This job takes a backup of the current ACS routines and replaces them with ones that include the changes added by the Sysplex Extensions.

Data class ACS routine

Note: Skip this step if you submitted job COPYACS in `SYSPLEX.PARALLEL.CNTL`.

Complete the following steps to update the data class ACS routine for the new DB2 subsystems that are provided with this package:

1. Save a copy of the current data class ACS member (ACSSTORD in this case) in case you need to fallback to the standard ACS routine.
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='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
```

These statements are in the ACSDB2DC member of `SYSPLEX.PARALLEL.CNTL`, so you can copy them from that 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 Sysplex Extensions 2020.

We found in our testing that the END statements in the ADCD-provided ACSSTORD member are incorrect. Review the member after you add the DB2 data class statements to ensure that every DO statement has a corresponding END statement.

Tip: The ACSSTORD member of `SYSPLEX.PARALLEL.CNTL` contains the corrected member that we used for our testing. If you have not changed the ADCD-delivered data class ACS routines, you could replace the ACSSTORD member in the `SYS1.SMS.CNTL` data set with the corresponding member from the `SYSPLEX.PARALLEL.CNTL` data set.

3. Press **F3** to save your changes.

Storage class ACS routine

Note: Skip this step if you submitted job COPYACS in `SYSPLEX.PARALLEL.CNTL`.

Complete the following steps to edit the ADCD-provided storage class ACS routine (DB2STORC):

1. Save a copy of the current storage class ACS member (DB2STORC in this case) in case you need to fallback to the standard ACS routine.
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’,‘CTS55’)
FILTLIST CICSRLS             INCLUDE(ZPDT.PST.**)
```

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

 WHEN (&DSN = &CICSRLS)
   DO
         SET &STORCLAS = 'VSAMRLS1'
         EXIT
   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.

Storage group ACS routine

Note: Skip this step if you submitted job COPYACS in `SYSPLEX.PARALLEL.CNTL`.

Finally, complete the following steps to edit the storage group ACS routine:

1. Save a copy of the current ACS 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’,‘CTS55’)
FILTLIST CICSRLS             INCLUDE(ZPDT.PST.**)
```

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
    WHEN (&DSN = &CICSRLS)
      DO
           SET &STORGRP = 'CICFILES'
           EXIT
      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.

Activating all your SMS changes

Having updated the ACS routines (either manually, or using the provided COPYACS job), you are now ready to translate them and then activate the new routines and the SMS base configuration changes. 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.SOW1.SCDS`.
4. Select option **2** (Translate) and press **Enter**.
5. Ensure that the SCDS Name is set to ‘`SYS1.SOW1.SCDS`’.
6. Set the ACS Source Data Set line to ‘`SYS1.SMS.CNTL`’.
7. On the ACS Source Member line, enter the name of your updated data class member (the default is ACSSTORD).
8. On the Listing Data Set line, enter the same member name as on the previous line (the data set will be created if it does not exist).
9. Press **Enter**.
10. Scroll to the end of the listing to ensure that the translation process ended with a return code of `0`.
11. Repeat these steps for the other ACS members that you updated (DB2STORC and DB2STORG).
12. Press **F3** to return to the primary ISMF menu.

13. Select option **8** (Control Data Set) and option **5** (Activate the CDS) to activate the modified CDS. Remember to place a forward slash (/) beside ‘Perform Activation.’ Press **Enter**.

The SMS-related changes are now complete.

4.3.15 Configuring the JES2 MAS

Important: The installation of the Sysplex Extensions 2020 makes only two changes that might affect your current ADCD system. The change that is described in this section is one of those changes. Therefore, you must be careful that this change does not conflict with any customization that you might have performed on your JES2PARM member and that the changes you make don’t inject any syntax errors.

JES2 must be changed to an MAS configuration with two members. This change does not affect the usability of the JES2 parameters for “normal” ADCD use.

The Sysplex Extensions 2020 package makes the following change to the JES2PARM member in the ADCD PARMLIB data set. Although it attempts to avoid making changes to the ADCD data sets, making changes in the `USER.Z24B.PARMLIB` data set necessitates several other changes, which increases the complexity with little benefit in return.

Also, the change made to the JES2PARM member is compatible with running the system in monoplex mode, which means that you can switch back and forth between monoplex and sysplex mode without making any further JES-related changes.

Tip: Because you are changing a member that is used by your running system, ensure that you 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 JES2PARM member.

To ensure that you have not introduced any syntax errors, restart JES2 at this point by using the following commands:

- `$PJES2,ABEND`
- Reply `END` to the $HASP198 message
- `S JES2,PARM=NOREQ`

The first time that you start JES2 after you make this change, you will see messages $HASP865 and $HASP870, which prompt you to confirm that you want to add member S0W2 to the JES2 MAS. Reply `Y` to this message.

JES2 Dynamic Proclib

While updating the JES2PARM, we took the opportunity to implement 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.Z24B.PROCLIB),
                DD(2)=(DSN=FEU.&SYSVER..PROCLIB)
                DD(3)=(DSN=ADCD.&SYSVER..PROCLIB),
                DD(4)=(DSN=CEE.SCEEPROC),
                DD(5)=(DSN=CSQ911.SCSQPROC),
                DD(6)=(DSN=IOE.SIOEPROC),
                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 (run the **`$D PROCLIB`** command), you can remove the PROC00 DD statements from the JES2 PROC.

The use of this capability delivers 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 JES2 proclib concatenation can now be changed dynamically, with no need to stop and restart JES2.
- If you have a syntax error in your JES2 proc JCL, JES2 will not start. However, if you have a syntax error in the PROCLIB definitions in the JES2PARM member, you are presented with a $HASP469 message.This message gives you the option of correcting or bypassing the invalid statement.

If a proclib concatenation (PROC00, for example) is specified in both the JES2 JCL and the JES2 parm member, the parm definitions are used rather than those 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.

4.3.16 Creating Parmlib members

Many of the attributes of how your system and sysplex work are controlled by using 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 are used for your base 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. If any parameter is not specified in IEASYSPS, the system uses 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) for our members. The members that we provide and the changes that are in each member are listed in Table 4-3.

Table 4-3 Parmlib members supplied by zPDT Sysplex Extensions 2020

Member	Supports concatenation?	Changed parms
AUTORPS	Yes	Added a rule to automatically reply to IGGN505A messages after 30 seconds.
BPXPRMBB/IZ/RZ/ZW/PS/01	Yes	► Point both systems at the same USERS file system. ► Changed multiple filesystem definitions to include &SYSNAME in the file name. ► Adjusted the AUTOMOVE settings of some file systems. ► Changed the MOUNT attribute from RDWR to READ for some file systems.
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. ► Added start for CFZCIM for z/OSMF.

Member	Supports concatenation?	Changed parms
CONSOLPS	No	Replaced entire member. Changes are to NAME parm on console definition, add the HOLDMODE(YES) parm, 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.
IEASYMBS IEASYMPS IEASYMPT IEASYMST	Yes	Replaced entire member. Adds IEASYSBS and IEASYSST sysparms.
IEASYSPS IEASYSBS IEASYSST	Yes	► Concatenated to IEASYS00. Overrides select statements 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.
IGDSMSPS	No	Start VSAM RLS and set sizes for RLS buffers.
IOEPRMPS	Yes	This member contains the parameters that are required by the zFS subsystem.
LOADPS LOADBS LOADST	No	Set appropriate IEASYM parameter.
LPALSTPS	Yes	Add CICS 5.5 to LPALST.
PROGPS PROGLD	Yes	Added SDSNEXIT library.
SHUTS0Wn	No	Add system-specific system shutdown commands for VTAMAPPL.
SMFPRMPS SMFPRMBS	No	Added log stream definitions.
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.Z24B.PARMLIB` data sets, submit the COPYPARM job in `SYSPLEX.PARALLEL.CNTL`.

Important: All of the Parmlib members that are described in this section are provided as part of this package. They *do not require manual changes* unless you must make an adjustment to allow for 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, skip to 4.3.17, "Network definitions" on page 67.

LOADPS member

This member is copied to the `SYS1.IPLPARM` data set rather than `USER.Z24B.PARMLIB`. However, it is included here because it is logically related to the members of the `USER.Z24B.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, LOADST, and other base sysplex-related members are described in 4.5.2, "Changes in Parmlib" on page 79 and 4.6.4, "Required Parmlib changes" on page 85.

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 when creating your LOADxx member.

IEASYMPS member

The original ADCD-supplied 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 more systems later, add sections that are based on these examples. For more information about the `IEASYMPS` member, see "Sample IEASYMxx member for sysplex" on page 126.

IEASYSPS member

A number of Parmlib members must be changed for the sysplex environment. Therefore, an `IEASYSPS` member was created that is concatenated to the standard `IEASYS00` member. The `IEASYSPS` member contains *only* the parameters that you need to override for the sysplex environment.

In addition to pointing at the other changed members, the `IEASYSPS` member 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 resources.

The standard ADCD system overrides the automatic startup of the z/OS Health Checker by specifying `HZS=*NONE` in the `IEASYS00` member. Because one of the objectives of this deliverable is to help vendors and customers create a more resilient environment that adheres to IBM's best practices here possible, the `IEASYSPS` member specifies that you start the Health Checker automatically and use the ADCD-provided `HZSPRMAD` member.

AUTORPS member

It is possible that the ADCD-provided Parmlib members might refer to data sets on volumes that you have not installed. This configuration can result in WTOR IGGN505A during NIP processing, and thus delaying the IPL until you see and reply to the message. To avoid this delay, an auto reply rule is added in the `AUTORPS` member to indicate that the system should reply with CANCEL to the IGGN505A message if no response is entered within 30 seconds.

Note that the auto reply rule does not have any knowledge of which data set or volume is missing. If you want to disable this rule, delete the `AUTOR=(00,PS)` line from the `IEASYSPS` member.

BPXPRMPS member

The ADCD-provided `BPXPRM01` member is copied into `BPXPRMPS` and includes several changes to reflect that the systems are running in a sysplex and to adhere to recommendations that are provided in *z/OS Distributed File Service zSeries File System Implementation z/OS V1R13*, SG24-6580. Specifically, the following changes are included:

- The `VERSION` file system is mounted read-only.
- The AUTOMOVE parameter was added to file systems that should be moved to the other system when the current owning system is stopped.
- The `USERS` file system is shared by the systems in the sysplex and uses AUTOMOVE.
- System-specific versions of the `SYSTEM`, `VAR`, `VARWBEM`, `ETC`, `USR.MAIL`, `WEB.CONFIG.ZFS`, `FELE20`, and `BGZ100` file systems were created for the S0W2 system.
- ZFS now uses an `IOEPRMPS` member created in `USER.Z24B.PARMLIB`. This member is pointed to from the IOEZPRM DD statement in the ZFS proc in `USER.Z24B.PROCLIB`.
- The z/OSMF file system mount point was moved from `/var/` to `/global/` in line with the latest guidance. The IZUSVR1 JCL was also updated to reflect this change.
- The Sysplex Extensions 2020 include a change to the BPXPRMPS member to make zFS run in the OMVS address space, rather than in its own address space. This change uses less CPU time than the normal method of defining zFS with an ASNAME parameter. In a zPDT environment, this can reduce the start time for z/OSMF by several minutes.

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 CLOCK00 member.

COMMNDPS member

The ADCD-provided COMMNDxx members start VTAM by using the ATCSTR00 member of `ADCD.Z24B.VTAMLST`. We did not want to change that member because 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 would be issued twice: Once from the COMMNDWS member and again from the COMMNDPS member. Therefore, we copied the contents of COMMNDWS into COMMNDPS, changed 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**. 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 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) and 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 also use the CTCs in a Parallel Sysplex, alongside the XCF CF structures if you want.

Note: Based on field experience, we increased the CLEANUP interval to 30 seconds in this release. This increase allows more time for partitioning of a system to complete successfully. Some customers were experiencing messages IXC405D or IXC420D when re-IPLing a system that was removed from the sysplex. The longer cleanup interval avoids these messages during a planned IPL.

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 practices 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.

IEFSSNPS

The new CICS and DB2 subsystems that are provided by the Sysplex Extensions must be defined to the system. This definition *could* be done dynamically by issuing a command after the system has initialized. However, we felt that it would be simpler to define a new IEFSSNPS member and concatenate that to the supplied IEFSSN00 member.

IGDSMSPS member

The IGDSMSxx member does not support the use of concatenation. Therefore, we copied the ADCD-provided IGDSMS00 member to a new IGDSMSPS member and added these parameters:

- Start the restartable PDSE address space (SMSPDSE1) in line with IBM's best practices. Enabling the restartable PDSE address space requires that PDSE Sharing is changed to EXTENDED, so we changed that parameter as well.
- Automatically start VSAM RLS at IPL time.
- Set the size of the buffers that can be used by VSAM RLS.
- Specify that VSAM data set of any CI Size can use VSAM RLS.

LPALSTPS member

Because we restored the volume for only one release of CICS (CICS TS 5.5), we created an LPALSTPS member to add the LPA data set for only that one release of CICS.

PROGxx members

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.

Note: The SHUTSYS started task automatically uses the SHUTSOWn member that is appropriate for the system where SHUTSYS is run. The SHUTS0W1 and SHUTS0W2 started tasks are no longer supplied.

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. We also added definitions for two SMF log streams *and* two of the new SMF Streaming in-memory buffers. These definitions are commented out until you are ready to use them.

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.

4.3.17 Network definitions

Before describing PROCLIB, TCP, and VTAM changes, this section describes the 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 have TCP set up 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.

We used VNC to log on to the Linux desktop remotely. (This VNC can be used to log on to VM from a remote window if necessary.) As a result, we did not find the absence of TCP on z/VM to be an inhibitor 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 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 116.
- 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 the corresponding 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 121 and Example A-5 on page 123.
- The `USER.Z24B.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 port name.
- The `USER.Z24B.TCPPARMS` data set contains the TCP PROFILE members (one per z/OS). The DEVICE and LINK statements use the port names 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 (&TRLNAM) 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 is as follows:

```
[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 for the `path` parameter on the `name awsosa` statement, run the Linux **`find_io`** command. The output from the command on our PC resembles the following example:

```
       Interface        Current         MAC              IPv4           IPv6
 Path   Name            State          Address          Address         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 port name 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 192.168.1.80 ETH2
```

You can see that the same port name (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 port names 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 192.168.1.90 ETH2
```

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

4.3.18 Creating PROCLIB members

A few changes are required to PROCLIB members for the base system. Many new procs are available for the Sysplex Extensions 2020-provided CICS regions and DB2 subsystems. Those changes are described in Chapter 6, “Sample CICSplex” on page 107, and Chapter 5, “Sample Db2 data sharing environment” on page 99. All changes and additions are made in the `USER.Z24B.PROCLIB` data sets.

Submit the COPYPROC job in `SYSPLEX.PARALLEL.CNTL` now to copy the MVS and CICS procs.

The DB2 procs are copied in a separate job that will only be executed if you follow the steps in 5.1, “Setting up the Sysplex Extensions 2020 Db2 data sharing environment” on page 100. For more information about the changes that we made, continue reading this section. Otherwise, you can skip to 4.3.20, “Creating TCPPARMS” on page 71.

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.Z24B.PROCLIB` because the ADCD-supplied VTAM00 member points at `ADCD.Z24B.PARMLIB` and we did not want to place any of our members in that data set. VTAMPS is started by running a command, such as **`S VTAMPS,M=&VTAMAPPL`**, where `&VTAMAPPL` is a system symbol that indicates which VTAMAPPL member of `USER.Z24B.PARMLIB` is to be 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 SHUTS0Wx 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.Z24B.PROCLIB` to point to the `USER.Z24B.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.Z24B.PROCLIB` data set) by the Sysplex Extensions 2020 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.

CICS procedures

The various CICS-related started tasks are described in 6.3, “Controlling your CICSplex environment” on page 111.

4.3.19 Grant Access to MVS Route Command

The **`MVS ROUTE`** command is run to start the various CICS regions. If you use the CICS support, the user ID that is associated with the started task that is used to start all the other CICS started tasks must be granted RACF access to the appropriate RACF class.

Run the PERMIT job in the SYSPLEX.PARALLEL.CNTL data set to issue the required RACF commands.

4.3.20 Creating TCPPARMS

Important: We mentioned in 4.3.15, "Configuring the JES2 MAS" on page 59 that there are two steps in the implementation of the Sysplex Extensions 2020 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 configuration. 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 will likely need to make adjustments 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.Z24B.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.Z24B.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
ALWAYSWTO 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
ALWAYSWTO NO
```

Submit the COPYTCPP job in `SYSPLEX.PARALLEL.CNTL` now.

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 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.Z24B.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, `192.168.1.80` and `192.168.1.90`, which means that both systems can use the same member.

If you use products that require IP name resolution (DB2 DDF or the TSO **`HOMETEST`** command, for example), use our sample GBLIPNOD member as an example of the changes that are required. We also updated the GBLIPDAT member to point it at the Google Name Server, using ‘`NSINTERADDR 8.8.8.8`’, and we changed the supplied GBLRESOL member to point at the GBLIPNOD member rather than the ZPDTIPN member.

Note: To be clear, we do *not* provide a job to replace the GBLIPNOD member of `ADCD.Znnn.TCPPARMS` with our version. If you require RESOLVER support, then you need to replace the ADCD-provided GBLIPNOD member of the `ADCD.Z24B.TCPPARMS` data set with your own version.

4.3.21 Creating VTAMLST members

We added several members to `USER.Z24B.VTAMLST`. We provide sample VTAMLST members in `SYSPLEX.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.

Change the members in the SYSPLEX.VTAMLST data set as required, and then submit the COPYVTAM job.

We also added members for CICS and DB2. Those changes are described in Chapter 6, “Sample CICSplex” on page 107, and Chapter 5, “Sample Db2 data sharing environment” on page 99.

4.3.22 Customization for Parallel Sysplex complete

You have now completed nearly all of the changes that are required to start the systems in Parallel Sysplex mode. Two steps are left that cannot be carried out until after the first system completes IPL in Parallel Sysplex mode. If this configuration is your target, proceed to 4.4, “Starting your Parallel Sysplex” on page 73. If your objective is to have a base sysplex under z/VM, some extra steps must be completed.

For more information about those steps, see 4.5, “Other steps to create a base sysplex under z/VM” on page 78. If your target environment is a base sysplex spread over two PCs, see 4.6, “Implementing a base sysplex without z/VM” on page 81.

4.4 Starting your Parallel Sysplex

Before you start the first system in Parallel Sysplex mode, shut down and then reload the z/OS system that is running under the BASEAD user ID. This process verifies that you still can fall back to monoplex mode if necessary. It should be possible to load your system in the same way as you did previously by using the same IPL address and load parm.

Important: Do *not* complete an IPL on the S0W2 system until you have completed the steps in this section.

After verifying that your current system still initializes successfully, the next step is to start your Parallel Sysplex:

1. Shut down the BASEAD system and log off that VM.
2. Run the following commands from the OPERATOR or MAINT user IDs to initialize the two coupling facility VMs:

   ```
   XAUTOLOG CFCC1
   XAUTOLOG CFCC2
   ```

   If you want to see the messages that are issued by the CFs, log on to the z/VM CFCONSOL user ID. 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 VM. 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 B4SYS1 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). You might also see an ILR030 PAGE DATASET MAY BE IN USE.....REPLY DENY OR CONTINUE message. This message is because a different userid (S0W1) is trying to use the page data set than was using it previously (BASEAD). You can reply **`r 00,CONTINUE`** to the message, assuming that the BASEAD system has been shut down.

   Sometime later, you might be prompted with WTOR IGGN505A, which indicates that a link library is not accessible. If you see that message and you do not need that data set, reply **`r 00,CANCEL`**. If you do not reply to that message, the z/OS Auto Reply function automatically replies cancel after 30 seconds.

Tip: The ADCD-provided PROGxx members might try to allocate data sets on volumes B4BLZ1, B4C541, and B4DBB1. If you did not restored one or more of those volumes, you receive the IGGN505A message. If you need the product that is associated with the missing volume, shut down the system (z/OS and z/VM), restore that volume, update your devmap file, and IPL the system again. If you do *not* need that product, the system automatically replies to that message after 30 seconds, and the IPL continues without that data set.

The system should not present any more WTORs.

When the IPL completes, move to one of the other z/VM screens and enter ‘DIAL S0W1’ on the COMMAND line to get the VTAM logon screen.

Defining required log streams

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

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 the job is run, the `DELETE LOGSTREAM` statement fails, but the other statements complete successfully.

Shut the system down at this point by running the **`S SHUTSYS`** command, then restart it. See 4.7.5, “Shutdown” on page 90 for more information about the recommended system shutdown process.

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

If you want to use LOGREC in DATASET mode, update the VTAMPS member to remove the SETLOGRC LS command. The ADCD-provided IEASYS00 member sets LOGREC to DATASET mode, so removing the SETLOGRC command from the VTAMPS member leaves it in DATASET mode.

If you want to use the SMF log stream, edit the SMFPRMPS member in `USER.Z24B.PARMLIB` and change the first line to say ACTIVE rather than NOACTIVE. The member is set up to run SMF in LOGSTREAM mode automatically as soon as you switch from NOACTIVE to ACTIVE. 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.Z24B.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.

Now all the log streams should be defined and in use. 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 4-2 are defined in your system.

Example 4-2 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, see 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 4-3.

In this case, the IXG007E message indicates that an error was encountered while processing the previous statement (#116). Also, 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 4-3 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
```

The CICS log stream staging and offload data sets are allocated in the CICS SMS storage group (CICS01). 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 (B4SYS1 and WORK01, if you use the provided VATLST00 and VATLSTPS members).

Define USS mount point for S0W2

The ADCD-provided system's UNIX file systems are designed for a single-system environment. Before you IPL the S0W2 system, submit the MKDIR job in `SYSPLEX.PARALLEL.CNTL`. This job performs the following functions:

- Creates a mount point for the system-specific file system for S0W2.
- Creates mount points for system-specific subdirectories for some program products on the S0W1 system.
- Mounts the SYSTEM file system for S0W2.
- Creates the mount points for the S0W2 system-specific subdrectories.
- Unmounts the S0W2 SYSTEM file system so that it can be mounted on S0W2 when you IPL that system.

IPL second system

You are now ready to start the S0W2 system, complete the following steps to start the second 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.

You might see some messages about XCF being unable to start the CTC paths to system S0W1. We included the CTC definitions in the COUPLEPS member to cater for base sysplex configurations. When running in Parallel Sysplex mode, those messages can be ignored.

You also notice several messages, as shown in the following example:

```
BPXF237I FILE SYSTEM ZFS.Z24B.VERSION 063
WAS ALREADY MOUNTED ON PATHNAME
/Z24B.
```

These messages are for file systems that will be shared between all the systems in the sysplex. All systems use a common BPXPRMPS member, so every system will try to mount those file systems when they are coming up. Because file systems can only be mounted on one system, the message is simply an informational message, indicating that system has already been mounted on another system in the sysplex. These messages can be ignored.

The first time that you IPL the S0W2 system, you may see IOE messages, saying that the system is waiting to see whether there is any other system using those file systems. You are getting those messages because the S0W2 file systems were created by copying the S0W1 file systems, so ZFS is waiting to ensure that S0W1 is no longer using those files. The ZFS start will pause for 65 seconds after each of those messages. However, you should only see these messages the first time you IPL S0W2. Subsequent IPLs will be a lot faster.

4.4.1 Parallel Sysplex implementation checklist

Because the implementation of the Sysplex Extensions 2020 Parallel Sysplex requires many steps and it is important that the steps are completed in the correct sequence, we provide the checklist in Table 4-4. You can use this checklist to track your progress and ensure that you do not accidentally omit any steps.

Table 4-4 Sysplex Extensions 2020 Parallel Sysplex implementation checklist

Task	Description	Done?
1	Install base ADCD z/OS system.	
2	Download and install the base ADCD z/VM system.	
3	Download and gunzip the Sysplex Extensions 2020 volumes.	
4	Create a backup of z/VM and z/OS volumes before making changes.	
5	Add Sysplex Extensions 2020 volumes to devmap.	
6	Add commands to devmap to start x3270 sessions.	
7	Add OSA definitions to devmap file.	
8	Restart zPDT and IPL z/VM.	
9	Update VM Directory to add the WRKALLEG keyword, if necessary.	
10	Log on to BASEAD and IPL z/OS in monoplex mode.	
11	Run IMPCONN job to IMPORT CONNECT user catalogs.	
12	Run DEFPAGE job to create S0W2 System VSAM data sets	
13	Run OMVSCLON job to create S0W2 copies of the OMVS file systems.	
14	Run ZFSLOGS job to create data sets for the zFS subsystem.	
15	Run HZSALLCP job to allocate the S0W2 Health Checker data set.	
16	Run RECATLG job to re-catalog the Sysplex Extensions 2020 CDSs and VSAM RLS Share Control Data Set in Master catalog.	
17	Run SMS0 jobs to update SMS config.	
18	Update ACS routines to assign new data, storage classes, and storage groups.	
19	Activate all SMS changes.	

Task	Description	Done?
20	Update MASDEF and PROCLIB statements in the JES2PARM member.	
21	Run COPYPARM job to copy Sysplex Extensions 2020 Parmlib members to `USER.Z24B.PARMLIB` and `SYS1.IPLPARM`.	
22	Run COPYPROC job to copy Sysplex Extensions 2020 Proclib members to `USER.Z24B.PROCLIB`.	
23	Run PERMIT job to grant access to the **`MVS ROUTE`** command to the START1 userid.	
24	Run COPYTCPP to copy TCP PROFILE and TCPDATA members to `USER.Z24B.TCPPARMS`.	
25	Run COPYVTAM job to copy VTAMLST members to `USER.Z24B.VTAMLST`.	
26	Reload BASEAD system in monoplex mode to ensure that everything still works as before.	
27	Logoff BASEAD.	
28	AUTOLOG CF VMs from CFCONSOL user.	
29	Log on to S0W1 and IPL by using the new Parallel Sysplex LOADPS member.	
30	Run the LOGREREP job to define the LOGREC log stream.	
31	Run the LOGROPR job to define the OPERLOG log stream.	
32	Run the LOGRRRS job to define the RRS log streams.	
33	Run the LOGRSMF job to define the SMF log stream.	
34	Run the LOGRHC job to define the Health Checker log stream.	
35	Run the MKDIR job to create zFS mount points for system S0W2.	
36	Log on to S0W2 and IPL.	

4.5 Other steps to create a base sysplex under z/VM

If your target environment is a base sysplex under z/VM, some other steps are required. There are also some messages that can be ignored when you start a base sysplex under z/VM. This section lists those messages and describes 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 definitions dynamically in each VM that is connected through the CTCs (S0W1 and S0W2 in this case) by running the VM **`DEFINE`** command.

However, we believe that it is less work in the long run 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 VMs):

```
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.

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 80.

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 VMs 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 in `SYSPLEX.PARALLEL.CNTL`.

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 running 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.Z24B.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.Z24B.PARMLIB` or IEASYSPS in `USER.Z24B.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.Z24B.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.Z24B.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 VMs 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 VMs are *not* logged on. To do this, 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 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 78. Before they can be used, you must connect the CTCs to each other. Although this connection must be done only from one of the VMs, 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 SOW2 E21
COUPLE E21 SOW2 E20
COUPLE E40 SOW2 E41
COUPLE E41 SOW2 E40
COUPLE E42 SOW2 E43
COUPLE E43 SOW2 E42
```

You are now 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`)

As with any time you load the first system in a sysplex, you might receive a WTOR (IXC405D) asking if you want to initialize the sysplex. If you see that message, reply **`R 00,I`**. 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, log on 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, 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.[8]

In the configuration that is shown in Figure 4-5 on page 82, 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 4-5, 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.

[8] The zPDT STP function requires zPDT GA6 or later.

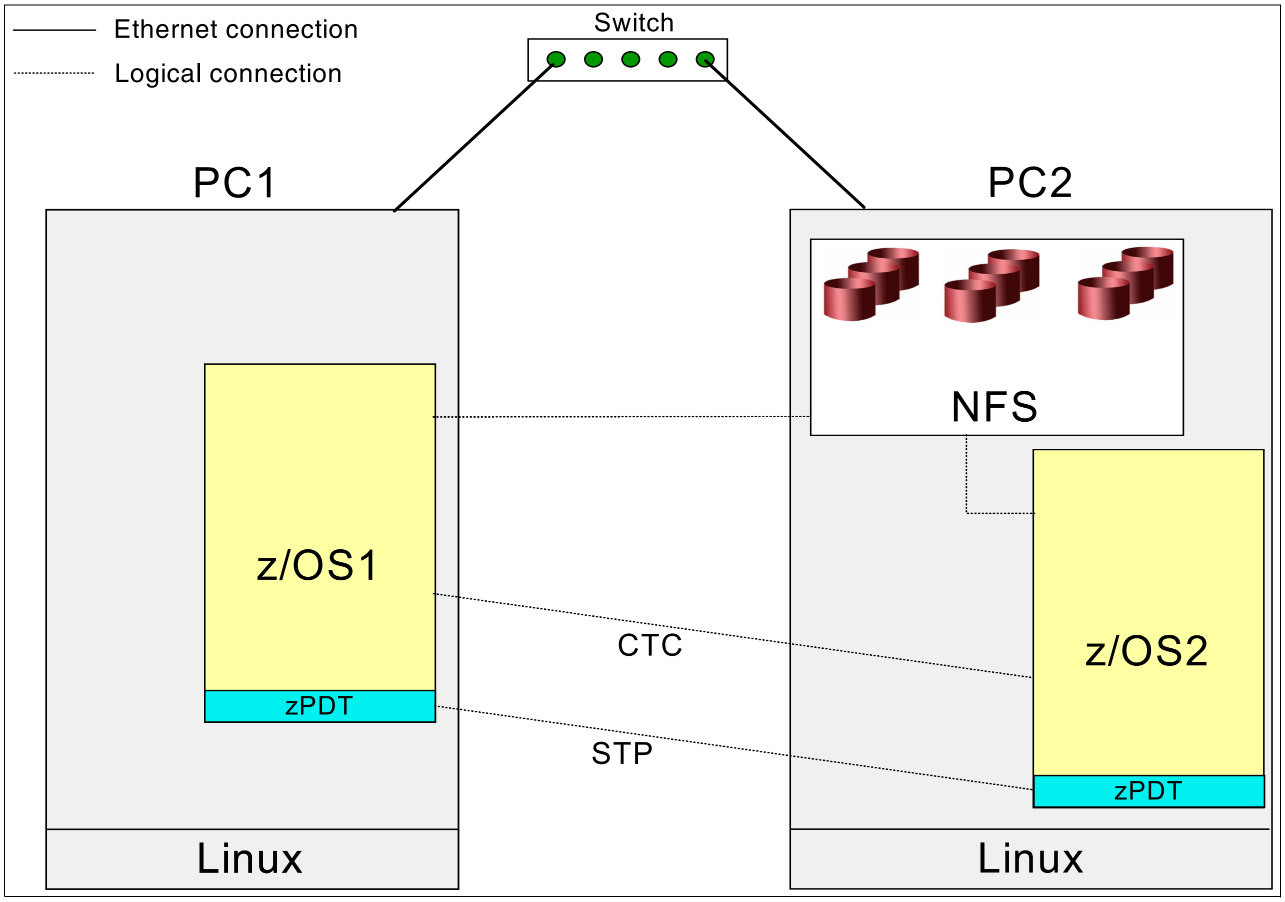


Figure 4-5 Base sysplex without z/VM

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: STP.

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.

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 4-4.

Example 4-4 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.

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
- The remote CTC to connect to

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. 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 4-5 and Example 4-6 on page 83.

Example 4-5 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 4-6 shows the devmap for PC2.

Example 4-6 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 80, the VM **COUPLE** commands connected CTC E40 in S0W1 to E41 in S0W2. XCF CTCs are unidirectional, so 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 CTC devices to the opposite type of CTC on the target system. In the statements that are shown in Example 4-5 on page 83, 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. However, 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 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 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 147.

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 Linux user ID, 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.Z24B.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.Z24B.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 CLOCKST member is a complete replacement for the CLOCK00 member.
- The CLOCKST member is copied into `USER.Z24B.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.Z24B.PARMLIB` when you run the STPPARM job.

VTAMBS

The VTAMBS member is similar to the VTAMPS member with the exception that it starts each RRS with a different group name. This technique avoids both RRSs trying to share the log stream (cross-system log stream sharing is not possible in a base sysplex).

Run the job STPPARM in `SYSPLEX.PARALLEL.CNTL` to copy these members into `USER.Z24B.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. Allow the first system to complete the entire startup process before you load the second system. Also, 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 network connection to the NFS server causes the IPL 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 will be much slower than the 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 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, 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` to allocate the DASDONLY log streams that are used by RRS on each system.

The OPERLOG log stream has a fixed name. 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, which negates much of the value of this function. As a result, we did not provide a job to create a DASDONLY version of the OPERLOG log stream.

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 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 still 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 cluttered window makes it easy to enter commands in the wrong 3270 session.

We suggest using two Linux desktop workspaces with the arrangement that is shown in Figure 4-6.[9] 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 4-6 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 151.

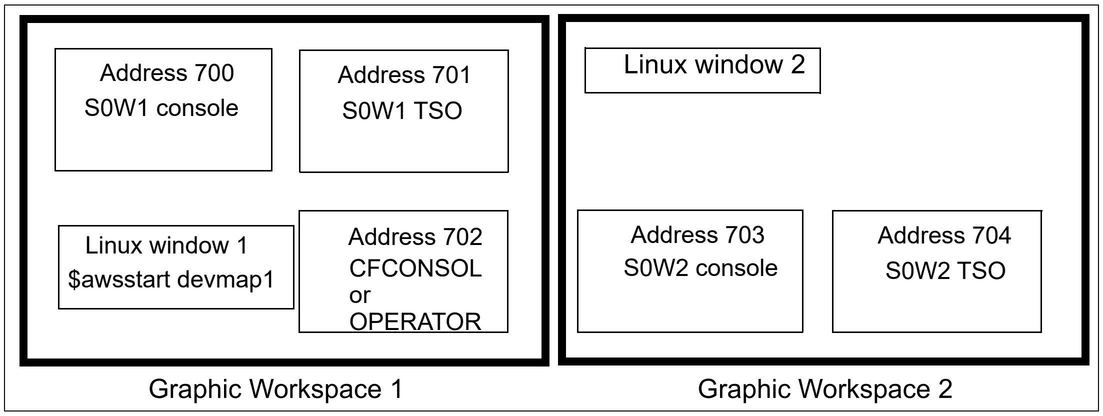


Figure 4-6 Sample layout for x3270 sessions

4.7.2 Post-IPL messages

If you switch back to running the S0W1 system in monoplex mode after it has been running in a multi-system sysplex, you might see unusual messages during start.These messages are issued 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.

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 Sysplex Extensions 2020 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.

[9] Another option is to use a separate PC for the 3270 sessions that are related to the S0W2 system.

- 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 Sysplex Extensions 2020 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 IPLing using a LOADxx member that indicates that it is to come up in base sysplex mode.

The SMFPRMBS 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 need to enable recording, you need to update the SMFPRMxx member to say ACTIVE rather than NOACTIVE.Then, issue the **`SET SMF=xx`** command to get SMF to use the parms in the named member. Note that you cannot use the **`SETSMF`** command to change the ACTIVE|NOACTIVE setting.

4.7.4 Sysplex policies

The sysplex policies that are provided by the Sysplex Extensions 2020 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. 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	Deletes and redefines the LOGREC log stream
LOGRHC	Deletes and redefines the z/OS Health Checker log stream
LOGROPR	Deletes and redefines the OPERLOG log stream
LOGRRRS	Deletes and redefines the RRS log streams
LOGRSMF	Deletes and redefines the SMF log streams

If you want to change the ARM, CFRM, or SFM policies, give the new policies a different name from 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 Sysplex Extensions 2020 are in Appendix B, “Sysplex couple data sets and policies” on page 133.

4.7.5 Shutdown

Although you might be tempted to shut down your system by simply killing it, this process can result in data loss and unexpected messages the next time the system is loaded. For these reasons, we 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 of the multiple levels of virtualization that are being used.

Buffering is used in both z/OS and 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.Z24B.PROCLIB`. It points at members SHUTS0W1 or SHUTS0W2 in `USER.Z24B.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 parentheses at the start of some lines refer to the device numbers that are used in Figure 4-6 on page 88):

```
        logoff of all TSO sessions
(703)   S SHUTSYS             (start shutdown script for SOW2)
        (IMPORTANT - wait for the $HASP250 ZFS PURGED and $HASP099 ALL
AVAILABLE FUNCTIONS COMPLETE messages - it might take a few minutes, and there
might be a pause in messages after VTAM stops and before you see the messages
about ZFS stopping.)
(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
        (Do NOT reply DOWN to this message. z/VM will automatically put SOW2
into a wait state when the vary offline processing has finshed)
        (wait for console activity to stop; usually after a CONSOLE
        PARTITION CLEANUP COMPLETE message.)
(703)   LOGOFF after the system enters a wait state
        (to log off the SOW2 virtual machine)
(700)   S SHUTSYS       (start shutdown script for SOW1)
        (IMPORTANT - wait for the $HASP250 ZFS PURGED and $HASP099 ALL
AVAILABLE FUNCTIONS COMPLETE messages - it might take a few minutes, and there
might be a pause in messages after VTAM stops and before you see the messages
about ZFS stopping.)
(700)   V XCF,SOW1,OFFLINE          (this allows PDSE, etc, to stop cleanly)
(700)   nn,SYSNAME=SOW1
(700)   LOGOFF after the system enters a wait state
        (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. During the shutdown processing for S0W1, you might see a number of BPX messages stating that the file system cannot be automoved. Those messages can be ignored. Because S0W1 is the last system in the sysplex, there is no alternate system available to move those file systems to. You might also see the following message:

```
BPX1070E USE SETOMVS ON ANOTHER SYSTEM TO MOVE NEEDED FILE SYSTEMS, THEN REPLY
WITH ANY KEY TO CONTINUE SHUTDOWN
```

This message is also related to the attempts to automove those file systems. Because the last system in the sysplex is being stopped, there is no other system to move the file systems to. Reply **Rnn,U** (or any character string) to the message to allow the shutdown to proceed.

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 VMs, 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
```

Another VM command that can be helpful is TERM HOLD OFF. VM sends a message to all logged on VMs at midnight informing them of the new time and date. If your z/OS console is logged on when this message is issued, it goes into a CP HOLDING state. If this state is not cleared, the console buffers will eventually fill and the system will go into a wait state. Issuing the TERM HOLD OFF command on the S0W1 or S0W2 console before you IPL z/OS stops the console going into the HOLDING state.

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). Then use the **`SETXCF ACOUPLE`** and **`SETXCF PSWITCH`** commands to copy the contents of the current CDSs to the new CDSs.
- Shut down the sysplex, bring up the original ADCD monoplex system (which has its own set of CDSs), delete the CDSs, define new (empty) data sets (by using the DEFCDSS job in `SYSPLEX.PARALLEL.CNTL` as a model), and bring the sysplex back up again.

Important: 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, such as 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 current 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.Z24B.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 System Logger CDSs, *all of the information in the existing log streams will be 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.[10]
2. Load the basic ADCD system (not under z/VM) with an IPL parameter, such as 0A82CS.
3. Using ISPF option 3.4, delete all of the Parallel Sysplex CDSs on volume CF0001.[11] 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 94. 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 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 4-7.

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

Example 4-7 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)
```

[10] 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 VMs after you stop and log off the S0W1 and S0W2 VMs.

[11] Our installation example in this chapter uses this volser, but other volumes can be used.

```
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 VMs, 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

    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:

```
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 IPL (with load member LOADPS), these log streams will be 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. 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              <==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 subsequent offload 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 **`alcckd`** command as described in 4.3.3, “Downloading and creating volumes” on page 44.
2. Add the new volume to the devmap. The devmap file 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 (initial password is ZVM710[12]).
   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 126. 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. Update the z/VM directory (run the **`directxa user direct c`** command).
3. Start one of the z/OS systems.

   Note: 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.

[12] This password is the password for the z/VM 7.1 ADCD system. Other z/VM implementations feature their own passwords.

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 connects 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 95. These SPECIAL connections are hardware connections between a 3270 session and the guest z/OS VM.
- 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 if you need more than 10 local 3270 sessions.
- 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[13] 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 a 10 GB zPDT system. The memory size is specified on the `memory` keyword. This value must be increased to accommodate 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 SAN Volume Controller 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 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. If you encounter these messages, add more paging volumes to z/VM before you add more work to z/OS.

[13] There is a limit of 30 because of TN3270 and VTAM definitions. These limits can be increased, if needed.

- Our CFs (in the z/VM directory) are defined with 5000 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. Generally, 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.
- SAN Volume Controller memory dumps are directed to B4SYS1 and B4SYS2. You might want to direct them to any 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 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. More PC memory, 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.

5

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 Sysplex Extensions 2020 and includes the following topics:

5.1 Setting up the Sysplex Extensions 2020 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. This package attempts to minimize the time and effort to get this additional functionality up and running.

If the package included the option of turning the ADCD-provided 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, minimal benefit is realized from the use of the Sysplex Extensions.

Therefore, the package provides a self-contained Db2 data sharing environment that is based on the Db2 V12 that is provided by ADCD. The Sysplex Extensions 2020 package consists of two new Db2 subsystems, a storage management subsystem (SMS) storage group that contains all of the Db2 databases, Db2 catalogs, logs, archive logs, ICF user catalogs that contain the entries for all these data sets, and so on.

A significant portion of the setup was completed as part of the standard Sysplex Extensions 2020 Parallel Sysplex implementation that is described in Chapter 4, “Installing the Sysplex Extensions 2020” on page 35. That is, much of the work is already done. Nevertheless, the sections that follow describe each of the implementation steps so that you can understand the changes that were made for Db2. It is made clear which steps were completed.

Note: Because the Db2 V12 IVP jobs reference a new buffer pool for 32K pages, we updated our CFRM policy in the POLCFRM1 member of SYSPLEX.PARALLEL.CNTL data set to add a DSNDPDG_GBP32K structure.

5.1.1 Overview of the Db2 environment

The following Db2 environment is delivered by the Sysplex Extensions 2020:

Db2 Data Sharing 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 names for tailored SDSNSAMP libraries:	`DSNDPDG.DPDx.SDSNSAMP` where *n* is 1 or 2 for DPD1 or DPD2 respectively.

5.1.2 Implementation

As a starting point for this process, it is assumed that you downloaded the DB2001 volume and that you completed all of the Parallel Sysplex implementation steps that are described in 4.3, “Parallel Sysplex set up steps” on page 41. Also, because this package is based on Db2 V12, the ADCD Db2 V12 volumes B4DBC1 and B4DBC2 must be available.

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 102. If you want to understand all of the steps that were necessary to prepare the sysplex for Db2 data sharing, continue on with the next section.

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 Sysplex Extensions 2020” 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, it is simpler to place all of the Db2-related data sets that are delivered by this package on a single volume in its own SMS Storage group. Configuring the package in this way makes the initial setup a little more complex but reduces the operational complexity and management on an ongoing basis.

The following changes to SMS are required for Db2:

- Add a Storage Group called `DB2FILES` and add a volume called `DB2001` to the storage group.
- Add a Storage Class called `PSADDON`.
- Add a Data Class called `DPDGDC`:
  - Set the Data Set Name Type to `EXTENDED`.
  - Set the Extended Addressability 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 Sysplex Extensions 2020” 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 Sysplex Extensions 2020” 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.Z24B.PARMLIB` contains definitions for two Db2 subsystems called `DPD1` and `DPD2`. In addition, it include two entries for the corresponding internal resource lock managers (IRLMs), called `DJD1` and `DJD2`.

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 PROGAD (APF) and PROGLD (LNKLST) members. So those members are included in the PROG concatenation in the IEASYSPS member. However, this package includes a PROGxx member for the SDSNEXIT library. That member is called PROGPS and is included in the PROG concatenation in IEASYSPS.

Important: All of the steps up to this point were completed during the standard Sysplex Extensions installation that is described in Chapter 4, "Installing the Sysplex Extensions 2020" on page 35.

The steps from this point forward were not completed; therefore, you must complete these steps.

RACF setup for Db2

Important: Some of the jobs in this step must run on the S0W2 system. Therefore, ensure that S0W1 and S0W2 systems are running before proceeding with this step.

Several RACF permissions are required to grant access to the new Db2 subsystems. Rather than you having to manually grant all of the accesses by using the RACF ISPF panels, jobs are provided in `SYSPLEX.PARALLEL.CNTL` to run the following required RACF commands:

1. Submit the job in the RACFDB21 member to create the RACF profiles used to protect various Db2 interfaces and permit IBMUSER to connect to Db2 from batch jobs.
2. Submit the RACFDB22 job to permit started tasks and batch jobs to connect to Db2. The job then refreshes the RACF profiles on system S0W1.
3. Submit the RACFDB23 and RACFDB24 jobs to add RACF profiles for the WLM environment needed for Db2.
4. Submit the RACFDB25 job to refresh the Db2 RACF profiles on system S0W2.

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.

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, adjust the sample jobs.

Workload Manager setup for Db2

Although they are not required for the basic Db2 environment that is provided by the Sysplex Extensions 2020, 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, to suppress those messages, submit the WLMDB21 job in `SYSPLEX.PARALLEL.CNTL`. This job defines and activates the application environments that are needed for this release's 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 Sysplex Extensions 2020 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* submit the DB2VTAML job in `SYSPLEX.PARALLEL.CNTL` now.

After the job completes successfully, run a **`RO *ALL,V NET,ACT,ID=DPD&SYSNUM.LU`** command to activate the Db2 VTAM definitions.

TCP setup for Db2

The section “Other network-related changes” on page 72 describes the RESOLVER function and the member that it uses to translate system names to IP addresses. If you do not change the GBLIPNOD member of `ADCD.Z24B.TCPPARMS` as described in that section, you see messages similar to the following example when you start Db2 on system S0W2:

```
DSNL512I  -DPD2 DSNLILNR TCP/IP GETADDRINFO(S0W2)
FAILED WITH
           RETURN CODE=1 AND REASON CODE=78AE1004
```

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 procedures for starting all of the Db2 tasks, `DSNDPDG.PROCLIB` contains the various Db2 procedures that are used for application development.

Submit the DB2PROCS job in `SYSPLEX.PARALLEL.CNTL` to copy the required members from `DSNDPDG.PROCLIB` to `USER.Z24B.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. This release of the zPDT Sysplex Extensions includes the CICS TS 5.5 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. It is hopeful that a future release of the Sysplex Extensions will provide this connection and a sample CICS/Db2 workload.

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; however, typically, the assumption is that DPD1 is running on system S0W1, and DPD2 is running on S0W2.

After Db2 starts successfully, first give the VTAMAPPL user ID shutdown authority. To achieve this, submit the GRANT job in `SYSPLEX.PARALLEL.CNTL`.

To stop Db2, run the following commands, which are included in the SHUTS0Wn members of `USER.Z24B.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. The IBMUSER and ADCDMST user IDs 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 (the DSNTIJRT and DSNTIJRV jobs, which enable the Db2 supplied routines).

If you want to change or refer to the Db2 setup jobs, they are contained in the `DSNDPDG.DPDx.SDSNSAMP` data set. DPD1 was the first subsystem, and 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 installation/migration clist to change the configuration, the output members from the initial setup are in the `DSNDPDG.DPDx.SDSNSAMP` data set. The member for DPD1 is called `DSNTIDXE`, and the member for DPD2 is `DSNTIDXF`. You must copy these members back to the ADCD-provided `DSNC10.SDSNSAMP` library before you can run the installation panels. Specify these members as the input on the initial panel.

The Db2 archive log data sets are defined in the Db2 ZPARM as having a retention period of 9999 days, so at some point you will need to clean them up to free up space. Complete the following steps:

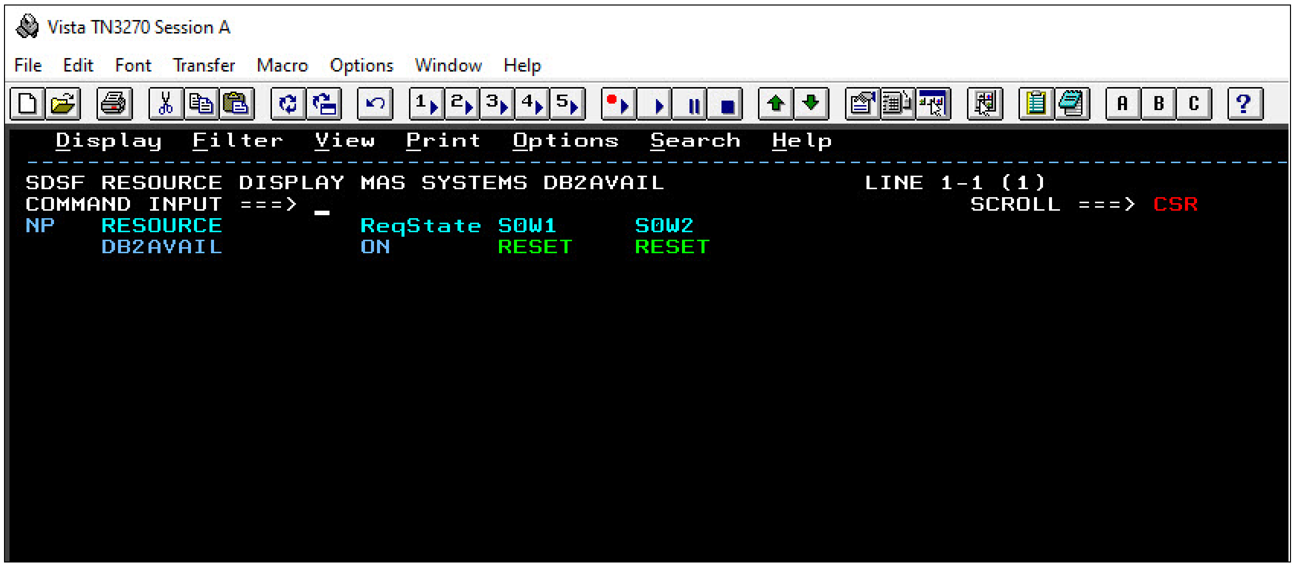
1. Do an orderly shutdown of both Db2s.
2. Run the DB2CLN1 job in `SYSPLEX.PARALLEL.CNTL` to get a list of the archive log data sets that are known to Db2.
3. Update and submit the DB2CLN2 job to delete the data sets from Db2’s bootstrap data sets.
4. Manually delete the archive log data sets by running **`IDCAMS DELETE CLUSTER`** commands. You must specify the PURGE option to override the unexpired retention period on the data sets.

5.4 Sample batch job to test Db2 data sharing

It is assumed that you have your own data and programs that you want to import into the data sharing group. However, if you first want to validate the correct functioning of Db2 and Db2 data sharing, this release of the zPDT Sysplex Extensions includes two jobs called `DB2IVP1` and `DB2IVP2` in the `SYSPLEX.PARALLEL.CNTL` data set. 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 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 is defined in the WLM policy. To use 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.



```
   Display  Filter  View  Print  Options  Search  Help
 -------------------------------------------------------------------------------
 SDSF RESOURCE DISPLAY MAS SYSTEMS DB2AVAIL            LINE 1-1 (1)
 COMMAND INPUT ===> _                                         SCROLL ===> CSR
 NP   RESOURCE            ReqState SOW1      SOW2
      DB2AVAIL            ON       RESET     RESET
```

Figure 5-1 SDSF Scheduling Environment Resource Status window

This example shows that the DB2AVAIL resource 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 connect 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.

6

Sample CICSplex

This chapter describes the CICSplex that is provided by the Sysplex Extensions 2020 and the steps that are required to get it up and running.

This chapter includes the following topics:

6.1 CICSplex overview

The standard ADCD system includes a CICS environment. However, it does not include sysplex-specific CICS functions. To address this requirement, the Sysplex Extensions 2020 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. It also includes support for VSAM Record Level Sharing and a sample application that exploits VSAM RLS in addition to the Named Counter and Data Tables servers.

Note: The Sysplex Extensions package does not use the standard ADCD-supplied CICS regions. The Sysplex Extensions-provided CICS environment is self-contained, with the exception of the CICS product data sets. It provides its own CICS regions, log streams, VSAM set sets, definitions, and so on. If you want to use the ADCD-provided CICS regions, follow the guidance provided in the ADCD documentation.

6.2 Setting up the Sysplex Extensions 2020 CICSplex environment

Implementing the Sysplex Extensions 2020-provided CICSplex consists of several steps. However, *all* 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 6.3, “Controlling your CICSplex environment” on page 111.

It is necessary to continue with this section *only* if you want to understand all of the steps that were necessary to add the CICSplex to the Parallel Sysplex.

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 Sysplex Extensions 2020” on page 35. This description is provided for your information only.

To minimize the time and manual effort to install the CICSplex, the CICS environment is packaged to be as self-contained as possible. The Sysplex Extensions 2020 includes a storage management subsystem (SMS) storage group that contains the user catalog that is used for all of the Sysplex Extensions-provided CICS regions. Specifically, it includes the runtime data sets, the CICS parameter library, CICS sample VSAM files, and the CICS log streams.

Because the CICS volume is SMS-managed, all data sets that are on that volume must have a storage class assigned to them. This prerequisite requires setting up an SMS storage class, a storage group (for the CICS SMS volume), and updates to the SMS ACS routines to assign the correct storage class and storage group to the new CICS-related data sets.

Specifically, you need to make the following changes:

- Add a storage group called `CICFILES` and a volume called `CICS01` to the storage group.
- Add two storage classes:
  - One storage class called `PSADDON`, which is the same storage class that is used by the DB2 SMS-managed data sets.
  - Another storage class called `VSAMRLS1`, which is the storage class for VSAM data sets that will be accessed in RLS mode.
- Update the SMS ACS routines. Sample filter list and SELECT section statements for the CICS data sets are contained in the ACSCICSC and ACSCICSG members of the `SYSPLEX.PARALLEL.CNTL` data set.

6.2.2 Parmlib changes for CICS

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 Sysplex Extensions 2020" on page 35. This description is provided for your information only.

A subset of CICS load modules must be placed in the link pack area (LPA). Instead of changing the LPALSTxx member in `ADCD.Z24B.PARMLIB`, a LPALSTPS member was created in `USER.Z24B.PARMLIB`. That member contains a single entry for CICS TS 5.5 and is concatenated to LPALSTPS in the IEASYSPS member.

To simplify the JCL for the CICS started tasks, a system symbol called `&CTSLEVEL` was created and set to 550. This symbol is defined in the IEASYMPS member.

CICS has two libraries (`DFH550.CICS.SDFHLINK` and `DFH550.CPSM.SEYULINK`) that are placed in the system LNKLST. The ADCD-provided PROGAC member contains the APF definitions for CICS TS 5.4 and 5.5, and the PROGLC member contains the LNKLST definitions.

If you downloaded only one of the two CICS releases provided by ADCD, you will receive an IGGN505A message during the IPL for the CICS LNKLST data set that cannot be found. To ensure that the IPL is not delayed by that message, an Auto Reply rule was added that replies "Cancel" automatically if you do not reply to the message within 30 seconds. If you want to speed up the IPL, you can change the wait time in the AUTORPS member of `USER.Z24B.PARMLIB`.

You need to add two MVS subsystems to z/OS for CICS:

- One MVS subsystem is for the CICS regions.
- One MVS subsystem is for use by the extra CICS servers (Named Counter Server, Temp Storage Server, and so on).

The Sysplex Extensions 2020-supplied IEFSSNPS member contains definitions for both of these subsystems.

CICS also requires a SAN Volume Controller. The ADCD-provided IEASVCCI member contains the definition for that SAN Volume Controller, so that member is concatenated to IEASVC00 in the 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 Sysplex Extensions 2020" on page 35. This description is provided for your information only.

Rather than having to provide separate VTAM definitions for each CICS region, model definitions are used. This method allows you 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 IMPORT CONNECT the CICS user catalog

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 Sysplex Extensions 2020" on page 35. This description is provided for your information only.

The entire Sysplex Extensions 2020 CICS environment is self-contained on one volume (CICS01). All data sets on those volumes are cataloged in the Sysplex Extensions-supplied `UCAT.CICSUSR` user catalog, and that user catalog is also contained on the CICS01 volume.

The 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 during the standard Sysplex Extensions installation process that is described in Chapter 4, "Installing the Sysplex Extensions 2020" on page 35. 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.

The LOGRCICS job in `SYSPLEX.PARALLEL.CNTL` adds these model definitions.

6.2.6 Copying CICS procs to USER.Z24B.PROCLIB

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 Sysplex Extensions 2020" on page 35. This description is provided for your information only.

A set of started task JCL for all the CICS regions is provided, including the CICSPlex SM address space (CMAS) and the web user interface (WUI). A master set of JCL (called *CICS1A*) is used for all the application-owning regions (AORs), and a different set is used for all the terminal-owning regions (TOR) (called *CICS1T*). The JCL for the individual AORs and TORs consists of just one statement, and that statement executes the corresponding CICS1A or CICS1T procedure. In an environment where all the regions are clones of each other, this process greatly simplifies the task of maintaining the CICS region JCLs.

All of these procedures were copied to `USER.Z24B.PROCLIB` by the COPYPROC job that was run as described in 4.3.18, “Creating PROCLIB members” on page 69.

6.3 Controlling your CICSplex environment

The CICSplex environment that is provided by the Sysplex Extensions 2020 consists of the following components:

- Two CICS Terminal Owning Regions on each system
- Two CICS Application Owning Regions on each system
- One CICS CMAS region per system
- One IBM CICSPlex web user interface (WUI) region for the sysplex
- One CICS CF Data Tables Server per system
- One CICS Named Counter Server per system
- One CICS Temporary Storage Server per system

Two TORs and two AORs are provided per system, because this configuration gives you the ability to stop one region (perhaps to apply a change) without affecting the ability of that system to process CICS transactions during that time.

The entire CICS environment can be started by running a single command (**`S STRTCICS`**), and stopped by running a single command (**`S STOPCICS`**). To make it easier to stop and start CICS, we provided the following procedures in `USER.Z24B.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

Initial start: Because no CICS log streams exist the first time you start each CICS region, the CICS JCL (the CICSWUI, CICS1A, CICS1T, and CMAS members of `USER.Z24B.PROCLIB`) are set up to specify `START=INITIAL`. This set up causes CICS to allocate new streams as part of its start. After the first successful start of each CICS region, update these four members to specify `START=AUTO` instead.

The full CICSplex consists of the following started tasks:

CICSCFDT	CICS CF Data Tables Server address space (same name is used on both systems)
CICSNCS1	CICS Named Counter Server address space (same name is used on both systems)
CICSTSSR	CICS Shared Temporary Storage address space (same name is used on both systems)
CICSWE11	CICS WUI region (1 per CICSplex)
CICS1A11	CICS AOR 1 on system S0W1

CICS1A12	CICS AOR 2 on system S0W1
CICS1A21	CICS AOR 1 on system S0W2
CICS1A22	CICS AOR 2 on system S0W2
CICS1T11	CICS TOR 1 on system S0W1
CICS1T12	CICS TOR 2 on system S0W1
CICS1T21	CICS TOR 1 on system S0W2
CICS1T22	CICS TOR 2 on system S0W2
CMASS0W1	CICS CMAS Region on system S0W1
CMASS0W2	CICS CMAS Region on system S0W2

The following naming convention is used for the CICS regions:

```
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 the CMASs are called *CMASS0W1* and *CMASS0W2*. If you have many CICS regions, you might want more than one CMAS per z/OS. In that case, 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* for two CMASs on system S0W1.

Suggestion: In line with the recommendation that the CICS CMAS regions be placed in the WLM SYSSTC Service Class, update 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 identify 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, the CICS regions are started by using a simple procedure 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 procedure in `USER.Z24B.PROCLIB` contains the following single JCL statement:

```
//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 parameters, including those for the CICS server regions, are in the `CTS55.USER.SYSIN` data set. The CICS servers use the following pool names:

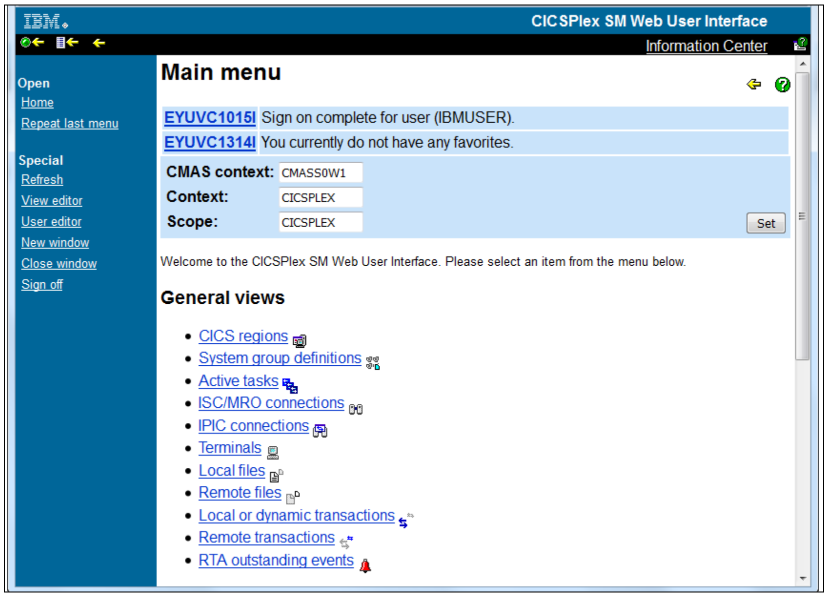
CF Data Tables	PDTCFDT1
Named Counter Server	PDTNCS1
Temporary Storage Server	PDTSTOR1

If you want to change the definitions of the CICS log streams or add new log streams, refer to the LOGRCICS job in the `SYSPLEX.PARALLEL.CNTL` data set. 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 (`START=INITIAL`). 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, 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. To logoff CICS, use the CICS **`CESF LOGOFF`** transaction.

After the WUI (started task is called *CICSWE11*) completes initialization, you can log on to it by browsing to `http://192.168.1.nn:3005` or `10.1.1.3:3005` if you are logging on from the same PC where your z/OS is running. When you connect, a window opens that is similar to the window that is shown in Figure 6-1.



Main menu

EYUVC1015I Sign on complete for user (IBMUSER).
EYUVC1314I You currently do not have any favorites.

CMAS context: CMASS0W1
Context: CICSPLEX
Scope: CICSPLEX

Welcome to the CICSPlex SM Web User Interface. Please select an item from the menu below.

General views

- CICS regions
- System group definitions
- Active tasks
- ISC/MRO connections
- IPIC connections
- Terminals
- Local files
- Remote files
- Local or dynamic transactions
- Remote transactions
- RTA outstanding events

Figure 6-1 CICS WUI Home page

If you are unfamiliar with the CICS WUI and perform much of the CICS administration work by using the old CICS 3270 interface, you are strongly encouraged to investigate the WUI as an alternative that is easier to use.

6.4 Migration considerations

In line with the process that is used by the ADCD deliverable, future releases of the Sysplex Extensions will provide full replacements for the CICS SMS volume (CICS01). If you want to add new CICS related data sets, 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 on a private volume (one that will not be replaced by a future ADCD or Sysplex Extensions release).
2. Create a high level qualifier for the new data sets and define that high-level qualifier as an alias, pointing at your new user catalog.
3. 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.

This appendix includes the following topics:

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 Sysplex Extensions 2020.

Note: We reduced the number of 3270 devices from 32 in the last release of Sysplex Extensions to 16. If you need the larger number, you can add definitions for devices 710 to 71F.

We also added a stanza to enable the zEDC PCIe card support in zPDT. At the time of this writing, the zEDC Accelerator feature support in zPDT is not yet available.

Example A-1 Sample devmap file

```
#Sample definitions for 2020 Sysplex Extensions package
[system]
memory 10000m
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

[manager]
name awsckd 27
device 0200 3390 3990 M01RES
device 0201 3390 3990 VMCOM1
device 0203 3390 3990 M01S01
device 0204 3390 3990 M01P01
device 0205 3390 3990 710RL1
```

```
[manager]
name awsckd 28
device 0a80 3390 3990 B4RES1
device 0a81 3390 3990 B4RES2
device 0a82 3390 3990 B4SYS1
device 0a83 3390 3990 B4USR1
device 0a84 3390 3990 B4CFG1
device 0a85 3390 3990 B4USS1
device 0a86 3390 3990 B4USS2
device 0a87 3390 3990 B4PAGA
device 0a88 3390 3990 B4PAGB
device 0a89 3390 3990 B4PAGC
device 0a8a 3390 3990 B4PRD1
device 0a8b 3390 3990 B4PRD2
device 0a8c 3390 3990 B4PRD3
device 0a8d 3390 3990 B4PRD4
device 0a8e 3390 3990 B4C551
device 0a8f 3390 3990 B4DBAR
device 0a90 3390 3990 B4DBC1
device 0a91 3390 3990 B4DBC2
device 0a92 3390 3990 B4ZWE1
device 0a93 3390 3990 SARES1

[manager]
name awsckd 29
device 0aa0 3390 3990 B4PAGX
device 0aa1 3390 3990 B4PAGY
device 0aa2 3390 3990 B4PAGZ
device 0aa3 3390 3990 B4SYS2
device 0aa4 3390 3990 CF0001
device 0aa5 3390 3990 CICS01
device 0aa6 3390 3990 DB2001
device 0aa7 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

# Define zEDC card
[pcipchid]
name awszedc 78 zedc
function 00c0 1
```

The numbers on the `name awsckd` statements are random. The only requirement is that they must be unique within that devmap file.

Run 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 120, Example A-4 on page 121, Example A-5 on page 123, Example A-6 on page 125, and Example A-7 on page 125 contain definitions for a VM user ID that acts as the owner of the z/OS volumes (MVSDUMMY), one non-sysplex guest (BASEAD), 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 The 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
   DASDOPT WRKALLEG
   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
   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 The 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 The VM Directory statements for S0W1

```
USER S0W1 S0W1 9000M 9000M 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
   SPECIAL 1400 MSGP CFCC1                                               12051039
   SPECIAL 1500 MSGP CFCC2                                               12051039
   CONSOLE 0700 3215 T                                                   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 The VM Directory statements for S0W2

```
USER S0W2 S0W2 9000M 9000M 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
   SPECIAL 1400 MSGP CFCC1                                                12051040
   SPECIAL 1500 MSGP CFCC2                                                12051040
   CONSOLE 0700 3215 T                                                    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 0AB0 0AB0 MW                                              12051040
   LINK MVSDUMMY 0AB1 0AB1 MW                                              12051040
   LINK MVSDUMMY 0AB2 0AB2 MW                                              12051040
```

```
   LINK MVSDUMMY 0AB3 0AB3 MW                                                  12051040
   LINK MVSDUMMY 0AB4 0AB4 MW                                                  12051040
   LINK MVSDUMMY 0AB5 0AB5 MW                                                  12051040
   LINK MVSDUMMY 0AB6 0AB6 MW                                                  12051040
   LINK MVSDUMMY 0AB7 0AB7 MW                                                  12051040
   LINK MVSDUMMY 0AB8 0AB8 MW                                                  12051040
   LINK MVSDUMMY 0AB9 0AB9 MW                                                  12051040
   LINK MVSDUMMY 0ABA 0ABA MW                                                  12051040
   LINK MVSDUMMY 0ABB 0ABB MW                                                  12051040
   LINK MVSDUMMY 0ABC 0ABC MW                                                  12051040
   LINK MVSDUMMY 0ABD 0ABD MW                                                  12051040
   LINK MVSDUMMY 0ABE 0ABE MW                                                  12051040
   LINK MVSDUMMY 0ABF 0ABF 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 The VM Directory statements for CFCC1

```
USER CFCC1 CFCC1 5000M 5000M 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 The VM Directory statements for CFCC2

```
USER CFCC2 CFCC2 5000M 5000M 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), an IEASYMPS member was created 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. The Sysplex Extensions 2020 IEASYMPS member is shown in Example A-8.

Example A-8 Sysplex Extensions 2020 provided IEASYMPS member

```
SYSDEF  SYMDEF(&UNIXVER='Z24B')
        SYMDEF(&CTSLEVEL='550')
        SYMDEF(&SYSVER='Z24B')
        SYMDEF(&SYSP1.='B4PRD1')
        SYMDEF(&SYSP2.='B4PRD2')
        SYMDEF(&SYSP3.='B4PRD3')
        SYMDEF(&SYSP4.='B4PRD4')
        SYMDEF(&SYSR2.='B4RES2')
        SYMDEF(&SYSS1.='B4SYS1')
        SYMDEF(&SYSC1.='B4CFG1')
        SYMDEF(&SYSNUM='&SYSNAME(4:1)')
        SYMDEF(&TMPSIZE='500')
        SYMDEF(&VTAMAPPL='VTAMPS')
        SYMDEF(&VTAMID='&SYSNAME(4:1)')
        SYMDEF(&TRLNAM='OSATRL&VTAMID.')
        SYSPARM(PS)  /* CONCATENATES IEASYSPS TO WHATEVER IS   */
                     /* SPECIFIED ON SYSPARM IN LOADXX MEMBER  */

SYSDEF  VMUSERID(S0W1)
        SYSNAME(S0W1)
        SYSCLONE(1A)
        SYMDEF(&VTAMSA='6')

SYSDEF  VMUSERID(S0W2)
        SYSNAME(S0W2)
        SYSCLONE(1B)
        SYMDEF(&VTAMSA='7')
```

Sysplex Extensions 2020 also provide two members for base sysplex mode:

- One member for running under z/VM (called *IEASYMBS*)
- One member 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 based 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 DEFPAGE member in the `SYSPLEX.PARALLEL.CNTL` data set allocates the system-specific data sets that are required by system S0W2. The JCL that is contained in DEFPAGE is shown in Example A-9.

Example A-9 Job DEFPAGE JCL

```
//DEFPAGE JOB CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID.
//S1      EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1      DD VOL=SER=B4PAGX,UNIT=3390,DISP=SHR
//DD2      DD VOL=SER=B4PAGY,UNIT=3390,DISP=SHR
//DD3      DD VOL=SER=B4PAGZ,UNIT=3390,DISP=SHR
//SYSIN    DD *
 DEFINE PGSPC(NAME(SYS1.S0W2.COMMON.PAGE) -
  FILE(DD1) CYLINDERS(100) VOLUME(B4PAGX))
 DEFINE PGSPC(NAME(SYS1.S0W2.PLPA.PAGE) -
  FILE(DD1) CYLINDERS(150) VOLUME(B4PAGX))
 DEFINE PGSPC(NAME(SYS1.S0W2.LOCALA.PAGE) -
  FILE(DD1) CYLINDERS(9745) VOLUME(B4PAGX))
 DEFINE PGSPC(NAME(SYS1.SOW2.LOCALB.PAGE) -
  FILE(DD2) CYLINDERS(9995) VOLUME(B4PAGY))
 DEFINE PGSPC(NAME(SYS1.SOW2.LOCALC.PAGE) -
  FILE(DD3) CYLINDERS(9995) VOLUME(B4PAGZ))
/*
//S3      EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1      DD VOL=SER=B4SYS2,UNIT=3390,DISP=SHR
//SYSIN    DD *
 DEFINE CLUSTER(NAME(SYS1.SOW2.MAN1) REUSE NIXD SPND SPEED SHR(2) -
  VOLUME(B4SYS2) CYLINDERS(40) RECSZ(4086,32767) CNVSZ(4096))
 DEFINE CLUSTER(NAME(SYS1.SOW2.MAN2) REUSE NIXD SPND SPEED SHR(2) -
  VOLUME(B4SYS2) CYLINDERS(10) RECSZ(4086,32767) CNVSZ(4096))
 DEFINE CL(NAME(SYS1.SOW2.STGINDEX) VOLUME(B4SYS2) CYLINDERS(4) -
  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.SOW2.LOGREC,UNIT=3390,VOL=SER=B4SYS2,
//        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` symbol 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.SOW1,PLPA.PAGE`, `SYS1,SOW1.COMMON.PAGE`, and so on. On system S0W2, the same statements define data sets `SYS1.SOW2.PLPA.PAGE`, `SYS1.SOW2.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, a subset of data sets must exist for each system. This job creates a copy of those data sets for the S0W2 system and places them on the B4SYS2 volume, as shown in Example A-10.

Example A-10 Cloning system-specific zFS data sets

```
//OMVSCLON JOB CLASS=A,MSGCLASS=X,REGION=OK,
//         NOTIFY=&SYSUID.
//*****************************************************************
//*
//*
//* CREATE SYSTEM-SPECIFIC ZFS DATASETS
//*
//*
//*****************************************************************
//********************************************************
//*
//*  ALLOCATE A SYSTEM ROOT FOR THE SOW2 SYSTEM
//*
//********************************************************
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(ZFS.SOW2.SYSTEM) -
            LINEAR CYL(2 1) SHAREOPTIONS(3,3)    -
            VOLUMES(B4SYS2))
/*
//********************************************************
//*
//*  NOW INITIALIZE THE NEW SYSTEM ROOT
//*
//********************************************************
//CREATE   EXEC   PGM=IOEFSUTL,REGION=OM,
// PARM=('format -aggregate ZFS.SOW2.SYSTEM')
//SYSPRINT DD     SYSOUT=*
//STDOUT   DD     SYSOUT=*
//STDERR   DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//CEEDUMP  DD     SYSOUT=*
//********************************************************
//*
//*  NOW RUN BPXISYS2 AGAINST NEW SYSTEM ROOT
//*
//********************************************************
//IKJEFT1A  EXEC PGM=IKJEFT1A,PARM='BPXISYS2 ZFS'
//SYSEXEC   DD   DSN=SYS1.SAMPLIB,DISP=SHR
//SYSTSPRT  DD   SYSOUT=*
//FSSYSTS   DD   DSNAME=ZFS.SOW2.SYSTEM,DISP=SHR
//SYSTSIN   DD   DUMMY
```

```
//*********************************************************
//*
//*  NOW USE DSS TO COPY THE SYSTEM-SPECIFIC ZFS FILES
//*
//*********************************************************
//CLONE   EXEC PGM=ADRDSSU,REGION=OM PARM='TYPRUN=NORUN'
//SYSPRINT DD SYSOUT=*
//FROMDASD DD UNIT=3390,DISP=SHR,VOL=SER=B4USS2
//TODASD   DD UNIT=3390,DISP=SHR,VOL=SER=B4SYS2
//SYSIN    DD *
 COPY DATASET(INCLUDE(                   -
                     ZFS.SOW1.DEV,                 -
                     ZFS.SOW1.ETC,                 -
                     ZFS.SOW1.TMP,                 -
                     ZFS.SOW1.USR.MAIL,            -
                     ZFS.SOW1.VAR,                 -
                     ZFS.SOW1.VARWBEM,             -
                     ZFS.SOW1.WEB.CONFIG.ZFS)) -
      LOGINDDNAME(FROMDASD) OUTDD(TODASD)  -
      TOLERATE(ENQF)  -
      RENAMEU(                                   -
         (ZFS.SOW1.DEV,ZFS.SOW2.DEV),                                  -
         (ZFS.SOW1.ETC,ZFS.SOW2.ETC),                                  -
         (ZFS.SOW1.TMP,ZFS.SOW2.TMP),                                  -
         (ZFS.SOW1.USR.MAIL,ZFS.SOW2.USR.MAIL),                        -
         (ZFS.SOW1.VAR,ZFS.SOW2.VAR),                                  -
         (ZFS.SOW1.VARWBEM,ZFS.SOW2.VARWBEM),                          -
         (ZFS.SOW1.WEB.CONFIG.ZFS,ZFS.SOW2.WEB.CONFIG.ZFS))       -
      SHARE ALLEXCP ALLDATA(*) CANCELERROR CATALOG
/*
//CLONE2  EXEC PGM=ADRDSSU,REGION=OM PARM='TYPRUN=NORUN'
//SYSPRINT DD SYSOUT=*
//FROMDASD DD UNIT=3390,DISP=SHR,VOL=SER=B4PRD2
//TODASD   DD UNIT=3390,DISP=SHR,VOL=SER=B4SYS2
//SYSIN    DD *
 COPY DATASET(INCLUDE(                   -
                     BGZ100.ETC.DBB.ZFS))     -
      LOGINDDNAME(FROMDASD) OUTDD(TODASD)  -
      TOLERATE(ENQF)  -
      RENAMEU(                                   -
         (BGZ100.ETC.DBB.ZFS,BGZ100.SOW1.ETC.DBB.ZFS))       -
      SHARE ALLEXCP ALLDATA(*) CANCELERROR CATALOG
/*
//CLONE3  EXEC PGM=ADRDSSU,REGION=OM PARM='TYPRUN=NORUN'
//SYSPRINT DD SYSOUT=*
//FROMDASD DD UNIT=3390,DISP=SHR,VOL=SER=B4PRD2
//TODASD   DD UNIT=3390,DISP=SHR,VOL=SER=B4SYS2
//SYSIN    DD *
 COPY DATASET(INCLUDE(                   -
                     BGZ100.SOW1.ETC.DBB.ZFS))     -
      LOGINDDNAME(FROMDASD) OUTDD(TODASD)  -
      TOLERATE(ENQF)  -
      RENAMEU(                                   -
         (BGZ100.SOW1.ETC.DBB.ZFS,BGZ100.SOW2.ETC.DBB.ZFS))       -
      SHARE ALLEXCP ALLDATA(*) CANCELERROR CATALOG
```

```
/*
//CLONE4  EXEC PGM=ADRDSSU,REGION=0M PARM='TYPRUN=NORUN'
//SYSPRINT DD SYSOUT=*
//FROMDASD DD UNIT=3390,DISP=SHR,VOL=SER=B4PRD4
//TODASD   DD UNIT=3390,DISP=SHR,VOL=SER=B4SYS2
//SYSIN    DD *
 COPY DATASET(INCLUDE(                  -
                     FELE20.SOW1.ETC.ZFS,         -
                     FELE20.SOW1.VAR.IDZ.ZFS, -
                     FELE20.SOW1.VAR.ZFS))        -
      LOGINDDNAME(FROMDASD) OUTDD(TODASD)  -
      TOLERATE(ENQF)  -
      RENAMEU(                               -
         (FELE20.SOW1.ETC.ZFS,FELE20.SOW2.ETC.ZFS),              -
         (FELE20.SOW1.VAR.IDZ.ZFS,FELE20.SOW2.VAR.IDZ.ZFS),      -
         (FELE20.SOW1.VAR.ZFS,FELE20.SOW2.VAR.ZFS))              -
      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

alcckd  B4C541 -ve
alcckd  B4CFG1 -ve
alcckd  B4DBC1 -ve
alcckd  B4DBC2 -ve
alcckd  B4DIS1 -ve
alcckd  B4DIS2 -ve
alcckd  B4PAGA -ve
alcckd  B4PAGB -ve
alcckd  B4PAGC -ve
alcckd  B4PRD1 -ve
alcckd  B4PRD2 -ve
alcckd  B4PRD3 -ve
alcckd  B4PRD4 -ve
alcckd  B4RES1 -ve
alcckd  B4RES2 -ve
alcckd  B4SYS1 -ve
alcckd  B4USS1 -ve
alcckd  B4USS2 -ve
alcckd  CF0001 -ve
alcckd  CICS01 -ve
alcckd  DB2001 -ve
alcckd  B4PAGX -ve
alcckd  B4PAGY -ve
alcckd  B4PAGZ -ve
alcckd  B4SYS2 -ve
alcckd  WORK01 -ve

alcckd  M01P01 -ve
alcckd  M01RES -ve
alcckd  M01S01 -ve
alcckd  VMCOM1 -ve
alcckd  710RL1 -ve
```

B

Sysplex couple data sets and policies

The standard ADCD deliverable includes several couple data sets (CDSs). The Sysplex Extensions 2020 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 Sysplex Extensions 2020 and includes the following topics:

- “ADCD CDSs” on page 134
- “Sysplex Extensions 2020 CDSs for Parallel Sysplex” on page 134
- “Job to create CDSs” on page 136
- “Sysplex Extensions 2020-provided policies” on page 139

ADCD CDSs

The ADCD system is delivered with the CDSs shown in Table B-1. The level of support is basic, as is appropriate 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.

Sysplex Extensions 2020 CDSs for Parallel Sysplex

Because of the Sysplex Extensions 2020' 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 Sysplex Extensions 2020 are listed in Table B-2.

Table B-2 Sysplex Extensions 2020-provided Parallel Sysplex CDSs

CDS type	Policy	Notes
ARM		Latest format level.
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=CFRM** command is shown in Example B-1. In this example, the highlighted lines show that the CDS was formatted to support the Asynchronous System Managed Duplexing Protocol.

Example B-1 Displaying the format of a CDS

```
D XCF,C,TYPE=CFRM
IXC358I  20.10.31  DISPLAY XCF 094
CFRM COUPLE DATA SETS
PRIMARY    DSN: PARALLEL.ADCDPL.CFRM.CDS01
           VOLSER: CF0001     DEVN: 0AA4
           FORMAT TOD         MAXSYSTEM
           10/04/2020 20:09:24        4
           ADDITIONAL INFORMATION:
            FORMAT DATA
             POLICY(4) CF(4) STR(100) CONNECT(8)
             SMREBLD(1) SMDUPLEX(1) MSGBASED(1) ASYNCDUPLEX(1)
ALTERNATE  DSN: PARALLEL.ADCDPL.CFRM.CDS02
           VOLSER: CF0001     DEVN: 0AA4
           FORMAT TOD         MAXSYSTEM
           10/04/2020 20:09:24        4
           ADDITIONAL INFORMATION:
            FORMAT DATA
             POLICY(4) CF(4) STR(100) CONNECT(8)
             SMREBLD(1) SMDUPLEX(1) MSGBASED(1) ASYNCDUPLEX(1)
CFRM IN USE BY ALL SYSTEMS
```

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

```
D XCF,POL,TYPE=CFRM
IXC364I  20.12.33  DISPLAY XCF 099
TYPE: CFRM
     POLNAME:       CFRM1
     STARTED:       10/03/2020 14:41:45
     LAST UPDATED:  10/03/2020 14:40:48
```

The source of the policies that are delivered in the Sysplex Extensions 2020 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 shows the name of the in-use policy.
2. The `SYSPLEX.PARALLEL.CNTL` data set contains a set of members called LSTxxx. Select the member 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)
         ITEM NAME(ASYNCDUPLEX) 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)
         ITEM NAME(ASYNCDUPLEX) 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)
##
```

Sysplex Extensions 2020-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 89. A copy of each policy is provided 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 Sysplex Extensions 2020 contains another CF structure compared to previous releases. The extra structure is required by the Db2 V12 IVP.

Also, the CFRM CDS was formatted to support the Asynchronous System Managed Duplexing capability that can be used by Db2 for its lock structure.

The CFRM policy is loaded in the CFRM CDS that is provided on the CF0001 volume; however, it is included 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(IRRXCF00_P001)
   INITSIZE(10M)
   SIZE(15M)
   PREFLIST(CFCC1,CFCC2)

   STRUCTURE NAME(IRRXCF00_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_GBP0)
   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)

    /* The GBP32K structure is used by Db2 for its 32K buffer pool*/
      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

```
//POLSFM1  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)
```

C

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 84 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 the sample configuration is not acceptable, this appendix describes a more robust alternative.

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.

Overview

The base sysplex is built with two Linux PCs, as shown in Figure 4-5 on page 82. This configuration is probably the simplest configuration possible, and it is used 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).

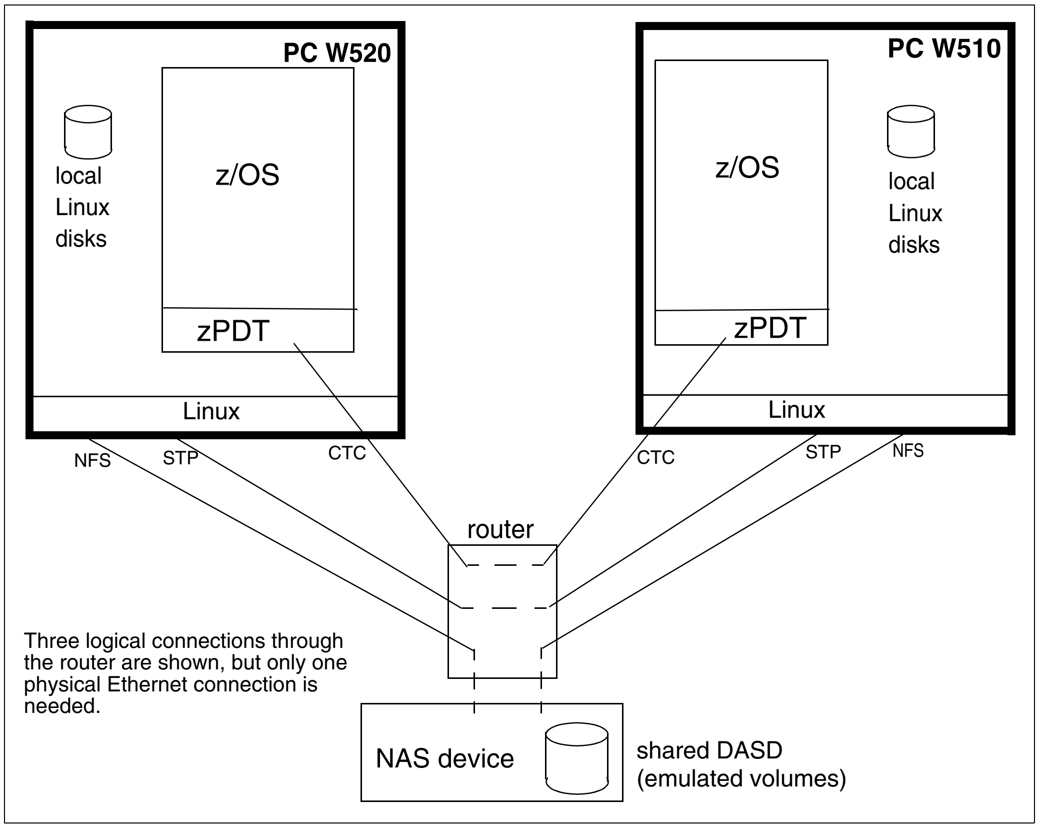

PC W520
local Linux disks
z/OS
zPDT
Linux
PC W510
z/OS
local Linux disks
zPDT
Linux
NFS
STP
CTC
CTC
STP
NFS
router
Three logical connections through the router are shown, but only one physical Ethernet connection is needed.
NAS device
shared DASD (emulated volumes)

Figure C-1 NAS storage for base sysplex

For this documentation, 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, the LAN connections were through 1 Gb switches or routers. All the 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.[1] 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 84.

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 that were used provided response times in the 0.15 millisecond range for simple **`ping`** operations.

[1] 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 unfamiliar 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 a base sysplex

If you are not familiar with z/VM or with running z/OS under z/VM, the commands that are shown in Example D-1 were run to start both systems under z/VM. This example is based on the use of two Linux workspaces (named `Linux1` and `Linux 2` in this example) that is described in 4.7.1, "Managing your x3270 sessions" on page 87. The parentheses at the beginning of each of the lines indicate which window is used for the command.

Example D-1 Commands issued to start both systems under z/VN

```
(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 4-6 on page 88)
              (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 SOW1      (password SOW1)
(700)         TERM CONMODE 3270
(700)         ipl 0a80 loadparm 0a82ps          (IPL SOW1 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 SOW1            (connect for TSO session)
(701)         (The VTAM logon screen should appear)
(701))     logon ibmuser                          logon to TSO)
(703)     logon to z/VM as user SOW2        (password SOW2)
(703)     TERM CONMODE 3270
(703)     ipl 0a80 loadparm 0a82ps              (IPL SOW2 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 SOW2              (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 7.1

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 Program 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.

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, run a **`ENABLE ALL`** command in the z/VM OPERATOR session.

E

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 associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG248386

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-8386.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
`syspext_8386-02_tar.gz`	Gzipped tar file containing all the Sysplex Extensions volumes

For ease of operation, download this file directly into the zPDT directory that holds all of your other zPDT z/OS volumes. Then, run a tar -xzvf command to extract the individual volume files, as described in 4.3.3, “Downloading and creating volumes” on page 44.

System requirements for downloading the Web material

The Web material requires at least 64 GB of Hard Disk Drive space.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics 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

Other publications

The following publications are also relevant as further information sources:

- *z/OS MVS Initialization and Tuning Reference*, SA23-1380
- *z/OS MVS Setting Up a Sysplex*, SA23-1399
- *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

IBM Global Services

ibm.com/services

Index

Redbooks

IBM zPDT Sysplex Extensions - 2020

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8386-02

ISBN 0738459143

Printed in U.S.A.