

IBM DS8000 Thin Provisioning

(Updated for Release 9.3)

Peter Kimmel

Connie Riggins

Jörg Klemm

Gauurav Sabharwal



Storage



International Technical Support Organization

IBM DS8000 Thin Provisioning

July 2022

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

Third Edition (July 2022)

This edition applies to IBM DS8000 Release 9.3, DS8000 License Machine Code (LMC) 7.9.30 (bundle version 89.30).

© Copyright International Business Machines Corporation 2017, 2022. All rights reserved.

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

Contents

Notices	v
Trademarks	vi
Preface	vii
Authors	vii
Now you can become a published author, too!	viii
Comments welcome	viii
Stay connected to IBM Redbooks publications	viii
Chapter 1. Thin provisioning	1
1.1 Overview	2
1.1.1 Thin-provisioning model	3
1.1.2 Thin-provisioning usage examples	4
1.2 DS8000-specific implementations	5
1.2.1 Extent Space Efficient thin-provisioned volumes	5
Chapter 2. Planning considerations and guidelines	7
2.1 Thin provisioning benefits	8
2.2 Planning considerations	9
2.2.1 Virtual capacity and metadata capacity	9
2.2.2 Over-provisioning	10
2.2.3 Extent pools	11
2.2.4 Out-of-space behavior	11
2.2.5 Sizing	12
2.2.6 Performance	13
2.2.7 Applications	14
2.3 Planning for ESE volumes as FlashCopy targets	15
2.3.1 Thin-provisioned ESE volumes for mirroring	15
2.3.2 Choosing an extent size	16
2.3.3 Setting up monitoring for capacity utilization	16
2.3.4 ESESizer tool	16
Chapter 3. Space-efficient volumes in the IBM DS8000	17
3.1 DS8000 logical volumes	18
3.1.1 Optional ESE repository for thin-provisioned logical volumes	20
3.1.2 Capacity allocation for ESE volumes	26
3.1.3 Out-of-space condition	27
3.1.4 Volume creation	28
3.1.5 Releasing space	29
3.1.6 The DS8900F flash backend	30
3.1.7 Migration considerations	30
3.1.8 Performance considerations	30
Chapter 4. DS GUI and DS CLI support for thin provisioning	31
4.1 DS GUI and DS CLI support to set up thin provisioning	32
4.1.1 Creating an extent pool with small extents by using the DS GUI	32
4.1.2 Creating an extent pool with small extents by using the DS CLI	33
4.1.3 Creating an extent pool with large extents	39
4.2 Creating ESE thin-provisioned volumes	42

4.2.1	Creating thin-provisioned volumes by using the DS GUI	42
4.2.2	Creating thin-provisioned volumes by using the DS CLI	43
4.2.3	Re-initializing online space-efficient volumes	45
4.3	Summary	46
Chapter 5.	Managing and monitoring thin-provisioned volumes	47
5.1	Management and monitoring capabilities	48
5.1.1	Options to reserve space and spare capacity	48
5.1.2	Overprovisioning control	48
5.1.3	Options to set thresholds and receive warnings	48
5.1.4	SNMP trap extent pool threshold	50
5.1.5	Managing out-of-space situations	51
Chapter 6.	Use of thin-provisioned volumes in IBM Z	53
6.1	Thin-provisioned Count Key Data devices	54
6.2	Advantages of using thin-provisioned volumes in z/OS	55
6.3	Migrating to thin-provisioned volumes	55
6.4	Thin provisioning support in IBM Z	56
6.5	DFSMSdss SPACEREL command	58
6.6	Thin-provisioning and copy services	60
Related publications	63
IBM Redbooks	63
Other publications	63
Online resources	63
How to get Redbooks publications	63
Help from IBM	63

Notices

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

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

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

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

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	IBM Spectrum™	System z®
DS8000®	Redbooks®	Tivoli®
Easy Tier®	Redpaper™	XIV®
Enterprise Storage Server®	Redbooks (logo)  ®	z/OS®
FlashCopy®	S/390®	
IBM®	System Storage®	

The following terms are trademarks of other companies:

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

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

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

Preface

Ever-increasing storage demands have a negative effect on an organization's IT budget and complicate the overall storage infrastructure and management. Companies are looking at ways to use their storage resources more efficiently.

Thin provisioning can help by reducing the amount of unused storage that is typically allocated to applications or users. Available for the IBM DS8000 for Fixed Block and Count Key Data (CKD) volumes, thin provisioning defers the allocation of actual space on the storage system until the time that the data must effectively be written to disk.

This IBM Redpaper™ publication provides an overall understanding of how thin provisioning works on the IBM DS8000. It also provides insights into the functional design and its implementation on the DS8900F and includes illustrations for the configuration of thin-provisioned volumes from the DS GUI or the DS CLI.

This edition applies to DS8900F Release 9.3 or later, yet with many aspects it is also usable with previous code levels or hardware generations.

Authors

This paper was produced by a team of specialists from around the world, working for the IBM International Technical Support Organization.

Connie Riggins is a DS8000 Copy Services and Copy Services Manager Subject Matter Expert with the DS8000 Product Engineering group. She started working at IBM in 2015. Before joining IBM, Connie worked at Amdahl Corporation (starting in 1991) as a Systems Engineer and later at Softek Storage Solutions as Product Manager of TDMF for z/OS.

Peter Kimmel is an IT Specialist and ATS team lead of the Enterprise Storage Solutions team at the EMEA Storage Competence Center (ESCC) in Frankfurt, Germany. He joined IBM Storage in 1999 and since then worked with all the IBM Enterprise Storage Server (ESS) and System Storage DS8000 generations, with a focus on architecture and performance. He was involved in the Early Shipment Programs (ESPs) of these early installations, and co-authored several DS8000 IBM Redbooks publications. Peter holds a Diploma (MSc) degree in Physics from the University of Kaiserslautern.

Jörg Klemm is a Senior Mainframe Consultant working for IBM Platinum Business Partner SVA System Vertrieb Alexander GmbH in Germany. He has over 20 years of experience in IBM working directly with customers, primarily focused on enterprise storage and specialized in Business Continuity solutions. His areas of expertise include Copy Services and Geographically Dispersed Parallel Sysplex (GDPS). Jörg has delivered GDPS solutions for over 15 years.

Gauurav Sabharwal joined IBM in 2009. He has approximately 18 years of experience in high-end file and block storage with different vendor products. He works in IBM LBS at the India location, and his areas of expertise include: Performance analysis, establishing high-availability and Disaster Recovery solutions, complex data migration, and implementation of storage systems. Gauurav holds a degree in information technology. He also acquired a Post-Graduate Program in Artificial Intelligence and Machine Learning from Texas McCombs business school.

Thanks to **Nick Clayton**, IBM UK, for his input and advice during the preparation of this paper.

Thanks to previous authors of this paper:

Bert Dufrasne, Andre Coelho, Peter Klee, Roland Wolf, Werner Bauer.

Now you can become a published author, too!

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

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

<https://www.redbooks.ibm.com/residencies>

Comments welcome

Your comments are important to us!

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

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

<https://www.redbooks.ibm.com/contact>

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

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

Stay connected to IBM Redbooks publications

- Look for us on LinkedIn:

<https://www.linkedin.com/groups/2130806/>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Thin provisioning

This chapter introduces the concepts and terminology that are related to thin provisioning. It includes a brief overview of the IBM DS8000 thin-provisioning implementation and the following topics:

- ▶ 1.1, “Overview” on page 2
- ▶ 1.2, “DS8000-specific implementations” on page 5

1.1 Overview

Thin provisioning is one way to help cut costs and reduce your storage footprint. Thin provisioning on volumes allows customers to create and use volumes where the sum of all virtual volume capacity might logically exceed the physical capacity of an extent pool.

However, the volumes allocate only the physical extents that they use, as needed. Thin provisioning allows you to define large volumes when they are created. This feature eliminates the need to go through the procedures to increase volume sizes later, which makes storage management easier.

Thin provisioning applies to shared storage environments. The concept of *thin provisioning* is to allocate storage (blocks or other elements of storage capacity) when application data is effectively written to a host-attached volume. This methodology is the fundamental difference with the traditional, fully provisioned volume, where the volume capacity is entirely allocated at the time the volume is created. In this case, the host to which the fully provisioned volume is attached owns the full capacity, which uses unused storage in the back-end system.

The general differences between full provisioning and thin provisioning are shown in Figure 1-1.

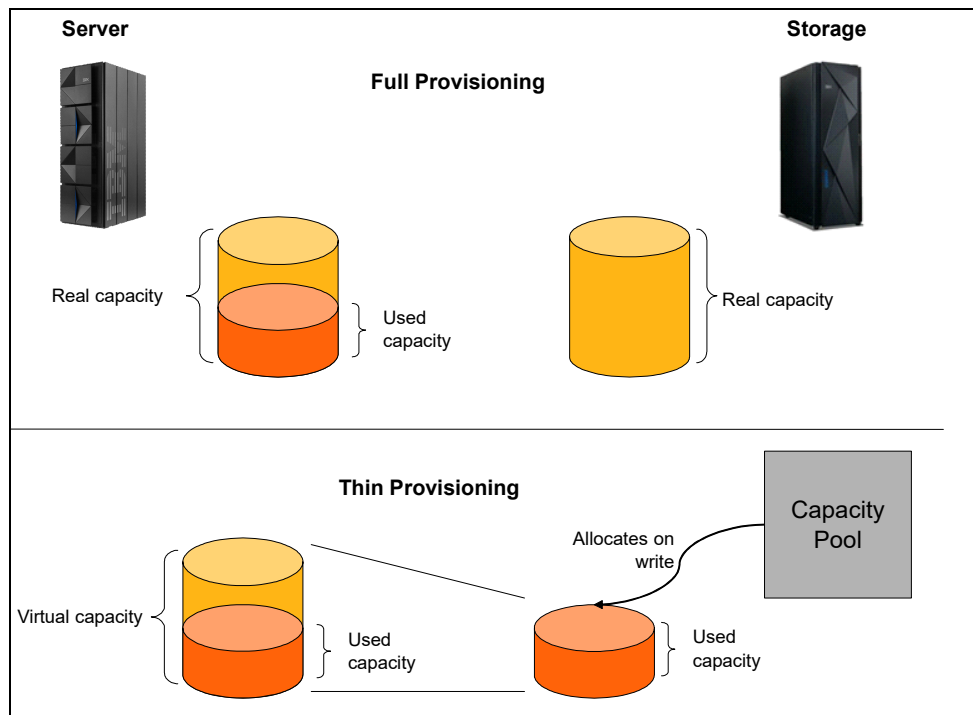


Figure 1-1 General differences between full provisioning and thin provisioning

A volume that supports thin provisioning is referred to as a *space-efficient* volume. When the volume is created, real capacity as required for metadata is physically allocated only. The DS8900F uses specific metadata to manage and determine when capacity must be allocated for write operations.

The volume is created with the full capacity that is requested by the user. However, this capacity is virtual because the allocation occurs only with the first host write to the volume. The sum of all space-efficient volume capacity represents the storage system virtual capacity.

When a space-efficient volume is assigned to a host, the host sees the whole (virtual) capacity of the volume as though it were a fully provisioned volume. All I/O activities that are performed by the storage system to allocate space when needed are fully transparent to the host.

The amount of real capacity in the storage system is shared between all space-efficient volumes. The amount of real capacity is, by definition, less than the total virtual capacity. The ratio between virtual capacity and real capacity represents the storage *oversubscription* or storage *over-commitment*.

Thin provisioning allows a more efficient capacity use in the storage subsystem. The automated capacity allocation (when required by applications) facilitates capacity management, as well. Capacity monitoring is still necessary, but can now be changed from a volume-based tracking, which can be done only on a per server basis to the back-end storage subsystem, where the overall view of virtual capacity and real capacity allocation is available.

1.1.1 Thin-provisioning model

The thin-provisioning model of the DS8900F (independent of any specific host platform) is shown in Figure 1-2. This section describes the elements of this common model.

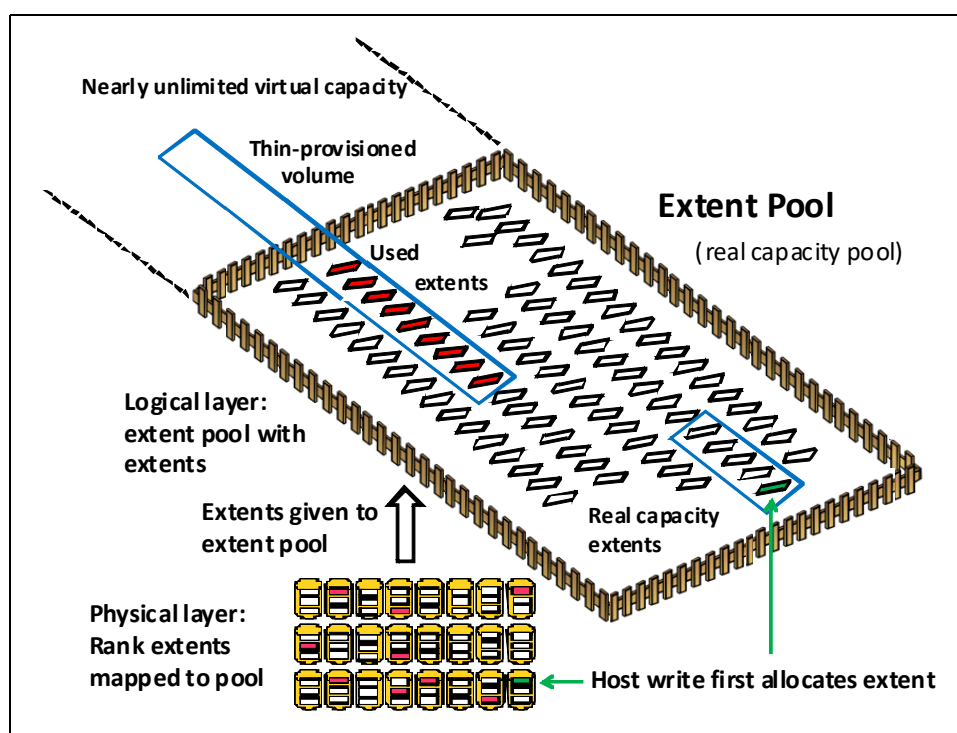


Figure 1-2 Common model of DS8900F thin provisioning

Virtual capacity

The *virtual capacity* is the sum of all defined host volumes capacity. The virtual capacity can be much larger than the physical capacity. Little real physical capacity is allocated when thin-provisioned volumes are created and only some capacity is allocated for the virtual volume's metadata. You can create as many space-efficient volumes as needed if the physical capacity is available for the metadata and you want some space for the actual user data.

Real capacity pool

The *real capacity pool* is the sum of all extents that are available in an extent pool in the DS8900F. It is the reservoir for physical capacity allocation when it is requested. When capacity allocation occurs, each allocation of real capacity is mapped to the space-efficient volume. The metadata for the virtual volumes also uses some of that capacity.

Space-efficient volumes

Space-efficient volumes are usable volumes that can be assigned to any host. These volumes do not have any physical capacity (except for their required metadata) when they are created. With the first write operation to the volume, real capacity from the real capacity pool is allocated to the volume.

1.1.2 Thin-provisioning usage examples

This section provides some typical usage examples of thin-provisioned volumes.

Thin provisioning for user data

As a simple example, consider typical user data in a business environment. Such data is normally the same kind for all users and might consist of local mail replica, cached data from web browsers, or other application data that is commonly used by users in a business environment.

In addition, most users have similar capacity requirements for this type of data. However, some users can also have specific applications with different capacity requirements. For example, users who frequently work with documents, spreadsheets, and presentations require less capacity than users who are working with images, other multimedia data, or other binary and scientific data.

Assuming that no user is requesting more than 10 GB of data and that the average of effectively used data is less than 6 GB per user, the storage administrator can decide to allocate 10 GB of virtual capacity for each user (as thin-provisioned volumes). However, in the back-end storage, the real capacity per user only amounts to 6 GB. If 100 users are assumed, the virtual capacity is 1 TB, but only 600 GB of real capacity is available in the storage subsystem. This approach is a valid only because the administrator knows from previous observations that not all users request the whole amount of physical capacity at the same time. Some request only 4 GB or less, and some might request the whole 10 GB. However, the assumption remains that on average they all request no more than 6 GB of real capacity.

It remains the responsibility of the storage administrator to monitor the allocation of real capacity to avoid any out-of-storage condition.

Thin provisioning in data centers

In data centers, a shared storage infrastructure provides storage for many different applications that are running on different servers. The initial storage capacity for those servers is often requested during their deployment phase. The requested capacity is most likely a combination of the initial required storage and an estimation of the growth over the estimated application lifetime.

Looking across all applications in a data center with focus on their storage capacity demand over time, experience shows that some applications grow quickly. However, other applications have almost no change in storage need, or do not even fully use their assigned initial capacity.

From an overall storage management perspective, this circumstance offers the opportunity to improve the use of the shared storage capacity. A policy can be established where assigned but unused capacity can be reassigned to applications that exhibit a more dynamic storage demand. To allow redistribution of capacity in the storage subsystem, the storage allocation must be separated from the storage configuration process. Thin provisioning offers this capability.

1.2 DS8000-specific implementations

The DS8000 supports Extent Space Efficient (ESE) volumes. The DS8900F treats every volume as an ESE volume regarding the metadata structure whether the volume is full-provisioned or thin-provisioned.

The user chooses between the following FB extent sizes for each extent pool:

- ▶ Large 1 GiB extents as used in previous implementations, which is still the default when you do not specify an extent size
- ▶ Small 16 MiB extents

For thin provisioning for CKD devices, the following extent sizes are available:

- ▶ Large extents that are based on 3390 Mod1 volumes with 1113 cylinders
- ▶ Small extents with 21 cylinders per extent

1.2.1 Extent Space Efficient thin-provisioned volumes

ESE volumes are thin-provisioned volumes that are designated for standard host access.

Thin-provisioned volumes are created with the DS CLI in either of the following ways:

- ▶ By specifying the space allocation method parameter **-sam ese** for the **mkfbvol** or **mkckdvol** commands
- ▶ By selecting **Provisioning: Thin Provisioned (ESE)**, as shown in Figure 1-3

For GUI versions before R9.3, the **Advanced (Custom) Settings** icon in the volume creation window allowed selecting the **Extent Space Efficient** option for the Allocation Settings.

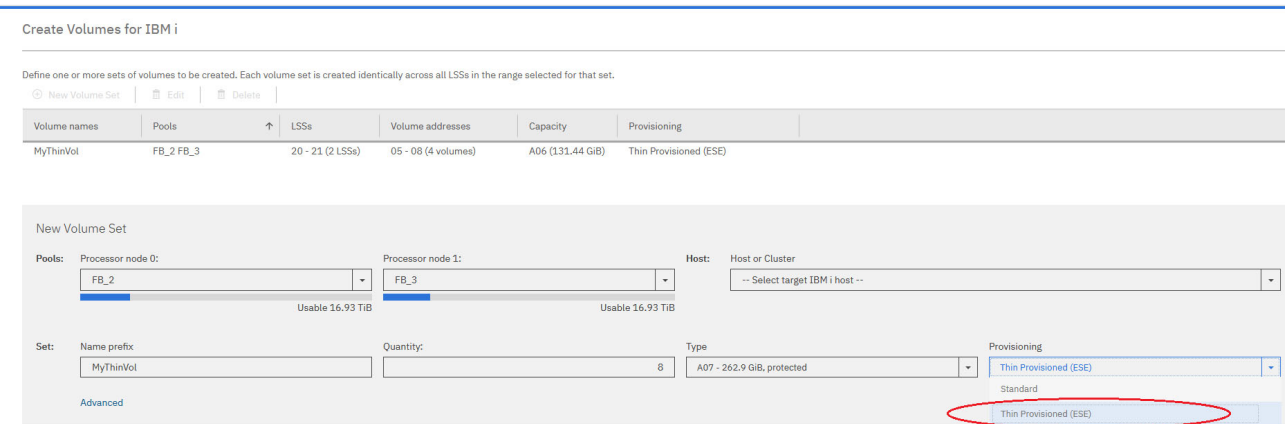


Figure 1-3 Creating space-efficient volumes using the DS GUI (code releases starting with R9.3)

The allocation of real capacity for ESE volumes occurs with writes to an unallocated address. Before the write is destaged from cache to disk, a new extent is dynamically allocated and assigned to the volume.

The extent must be formatted after the extent is allocated. The extent initialization is done by using the Quick Init process that was introduced with the DS8000 microcode Release 4.3.

The storage administrator has the choice to create extent pools of different extent sizes. The supported extent sizes are:

- ▶ Fixed Block volumes: 1 GiB or 16 MiB
- ▶ Count Key Data volumes: 1113 cylinders or 21 cylinders

An extent pool cannot have a mix of different extent sizes.

For thin-provisioned volumes, small extents provide a much better capacity utilization. However, small extents cause more management overhead for the storage system because more extent allocations occur compared to large extents. At allocation time, the write performance might be slightly affected.

The response time for the first write operation might increase, but not significantly. After the extent is allocated and formatted, the ESE volume performs the same as standard logical volumes.



Planning considerations and guidelines

This chapter presents general guidelines and describes important considerations for using the thin provisioning functionality. The information applies to the thin-provisioning feature that is available for the IBM DS8900F.

This chapter includes the following topics:

- ▶ 2.1, “Thin provisioning benefits” on page 8
- ▶ 2.2, “Planning considerations” on page 9
- ▶ 2.3, “Planning for ESE volumes as FlashCopy targets” on page 15

Terminology: This chapter uses the term *thin provisioning* as it relates to Extent Space Efficient (ESE) implementation only.

2.1 Thin provisioning benefits

Thin provisioning offers the following typical advantages:

- ▶ Reduced storage management efforts:
 - Use of over-provisioning in file systems, databases, and logical volume managers without the underlying physical capacity to be present.
 - Real capacity can be added transparently to the software stack later, as required.
- ▶ Improved capacity utilization:
 - Physical capacity is not allocated until it is used.
 - Unused capacity for a set of volumes is in a shared pool of overall available capacity.
 - Contingency capacity can be shared between this set of volumes. The average contingency capacity per volume can be less compared to the case in which it cannot be shared and every volume includes allocated contingency capacity. This shared contingency capacity leads to higher utilizations of the physical storage capacity; therefore, acquisitions of more physical capacity might be reduced or deferred to a later time.
 - By sharing the capacity in a pool that is used to provision a set of space-efficient logical volumes, deviations in allocation requirements of specific volumes tend to average out over the whole capacity pool. This change increases the amount of tolerable over-commitment and makes it more predictable.
 - The limit of tolerable over-provisioning without running into out-of-space conditions depends on the following circumstances:
 - Characteristics of the applications that are using the space-efficient volumes
 - Granularity of incremental capacity allocations to the space-efficient volumes
 - Rates at which capacity is used versus being released

A tradeoff exists between performance and space-efficiency, which depends on the granularity of capacity allocation. Consider the following points:

- ▶ A fine granularity leads to the best space efficiencies, but might introduce some performance overhead for handling and updating more metadata and when capacity allocations are performed on a small scale.
- ▶ Although a coarse granularity might be less space efficient for the applications, it tends to have less affect on performance because of the following circumstances:
 - Larger capacity is allocated at one time; therefore, fewer allocations are necessary for a certain amount of data.
 - Reduction in metadata accesses to determine whether capacity is allocated.
 - Fewer updates are required to the metadata to reflect the changes in allocations.
 - More friendly for sequential workloads because sequential logical tracks are grouped into fewer capacity allocations.

In the DS8900F, thin provisioning is implemented with a choice between the following extent sizes:

- ▶ Large extents (1 GiB [FB], 1113 cylinders [CKD])
- ▶ Small extents (16 MiB [FB], 21 cylinders [CKD])

In both implementations, the metadata space overhead is taken from the extent pool where the user data also is stored.

Quick Init is a function that allows a fast extent initialization process. This quick initialization helps alleviate any performance implication that results from the overhead that is introduced by the nature of thin provisioning.

Quick Init also is designed to start Copy Services operations directly after invocation, so you do not need to wait for the initialization process to finish.

2.2 Planning considerations

Specific elements to consider when you plan to use thin provisioning are described in this section.

2.2.1 Virtual capacity and metadata capacity

In previous DS8000 firmware releases (before R8.1), virtual capacity was sized and configured before ESE volumes were created. This requirement is no longer needed for ESE volumes with large extents and it is not possible at all for volumes with small extents.

When many large thin-provisioned volumes are defined, you must understand the way metadata is allocated. Other types of metadata are available in the DS8900F. This section describes the volume metadata, which is in the extent pool with the user data.

A distinction must be made between extent pools with small and with large extents.

Size of metadata

For extent pools with small extents, metadata blocks are one extent.

- ▶ FB pools have 16 MiB extents.
- ▶ CKD pools have 21 cylinder extents.

For extents pools with large extents:

- ▶ For FB, a 1-GiB extent is subdivided into 16 MiB subextents.
- ▶ For CKD, a one 1113-cylinder extent is subdivided into 21-cylinder subextents.

Fixed Block

Every FB volume that is created requires one 16 MiB block for metadata. In addition to this basic metadata block for the volume, every 10 GiB of virtual capacity of an FB volume needs another 16 MiB metadata block. For example, a 100 GB FB volume requires 11 metadata extents: one for the volume and 10 for the capacity.

Count Key Data

Every CKD volume that is created uses a 21-cylinder metadata block. Every 11130 cylinders (or 10 Mod-1's) of virtual capacity requires a 21-cylinder metadata block. For example, a 3390 Mod-27 (32760 cylinders) requires four metadata extents: one for the volume and three for the capacity.

Therefore, for extent pools with large extents, metadata is allocated in large extents. The large extent is filled with small subextents as needed until the allocation of another large extent for metadata is required, as shown in Figure 2-1 on page 10.

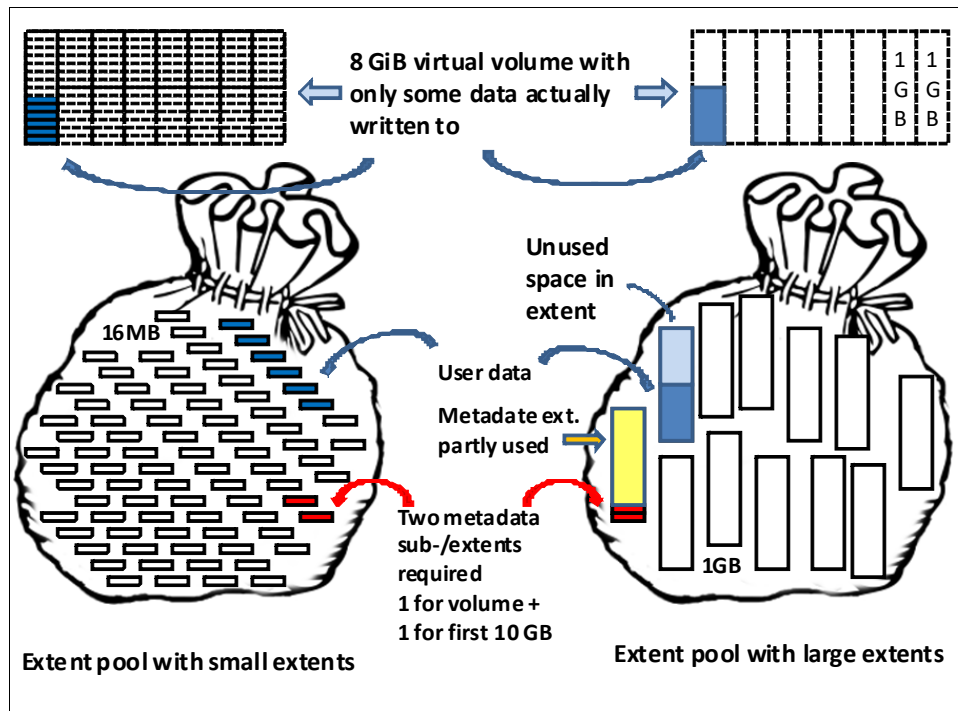


Figure 2-1 Extent pools with small and large extents

2.2.2 Over-provisioning

Over-provisioning is the ratio of virtual capacity to real capacity. The virtual capacity is the amount of storage that is committed to the applications. The real capacity is the installed hardware capacity.

Over-provisioning is a key consideration in a thin provisioning environment. To size the real capacity pool, which translates into real storage hardware investment, the sustainable over-commitment or over-provisioning must be determined. Although over-provisioning allows you to save or defer the cost of hardware investments, it also introduces the risk of running out of real capacity in a running production environment.

A generic recommendation about sustainable over-provisioning ratios is difficult to give because the workloads and client requirements vary broadly. You should monitor what storage is used, what storage remains unallocated, and how this usage changes over time. By doing so, the storage administrator gains a better understanding of which volumes tend to under-utilize their configured size. This allows a better estimation of the amount of over-commitment that is sustainable in this environment.

The storage administrator must have an understanding of the effect of running out of space in their environment and what actions must be taken to recover from the situation. The DS8900F provides options to reserve a capacity buffer and to set thresholds for monitoring purposes, as described in Chapter 5, “Managing and monitoring thin-provisioned volumes” on page 47. Overprovisioning is managed through a ratio limit that can be set and enforced by the system, as described in “Support for overprovisioning controls” on page 37.

The tradeoff between cost savings and risk of outage is controlled by the over-provisioning factor. This value must be obtained carefully. The following procedure is suggested to help you determine an acceptable over-provisioning value:

1. Perform an analysis of the current storage utilization on a per server or application basis. Use the ratio of the assigned capacity to the used capacity as a first general indication for a possible over-provisioning value.
2. Create an estimate of future capacity needs to determine whether the over-provisioning value that was obtained in step 1 will remain constant over time. Perform a review of the storage demands history across all applications that are candidates for thin provisioning. The outcome of this review is an indication of how to adjust the over-provisioning ratio.

For example, if many applications in your environment feature low or moderate storage growth and only a few applications with dynamic storage growth, you can adopt a more aggressive over-provisioning value. This is because the demands of the few dynamic applications can be supplied by a huge base of applications with unused storage capacity.

Every application is driven by the business for which it was designed. In this sense, knowing the expected development of the business behind that application also gives some indication of the future storage demands. If a high growth of storage demand is expected, it might be appropriate to reduce the over-provisioning ratio to decrease the risk.

2.2.3 Extent pools

In a mixed extent pool (that is, one with standard volumes and ESE volumes), the available free extents can be allocated to either type of volumes. However, every new standard volume that is created is reducing the number of free extents, which increases the over-provisioning ratio for the ESE volumes. From a management perspective, it is likely desirable to have standard volumes and ESE volumes in separate extent pools.

Minimizing the number of extent pools is maximizing the options to share the contingency capacity between the volumes inside the individual extent pools. This configuration helps avoid out-of-space conditions in one extent pool although other extent pools might still have enough contingency or buffer capacity available.

The DS8000 thin-provisioning implementation offers the flexibility to have standard volumes and thin-provisioned volumes in the same extent pool, which provides a choice (unlike other implementations). A recommendation to have separate extent pools for the two volume types is based on an ease-of-management perspective.

Merging extent pools: The DS8000 provides the capability to merge two extent pools, if wanted. However, only extent pools of the same extent size, and affinity to the same processor complex, can be merged.

2.2.4 Out-of-space behavior

Over-provisioning or over-commitment of capacity can result in a situation where the needed capacity exceeds the capacity that is still physically available. In the DS8000 implementation of thin provisioning, a write inhibit is sent to the host for any write operations that exceed this amount, which often causes the software to fail. The DS8000 rejects write accesses to the affected volumes until the required added real capacity is made available for allocation.

It is important to determine the tolerable over-provisioning ratio for a specific environment as a realistic, stable average over time and use the threshold mechanisms that are provided to receive warnings ahead of time. For this purpose, the thresholds must be set according to the individual client needs that are driven by the workload and application characteristics.

Tip: An overprovisioning ratio limit can be set and enforced by the system, as described in “Support for overprovisioning controls” on page 37.

Setting the overprovisioning ratio changes the system behavior to prevent an extent pool from exceeding the overprovisioning ratio in the following situations:

- ▶ Prevent volume creation, expansion, or migration
- ▶ Prevent rank depopulation
- ▶ Prevent pool merge
- ▶ Prevent turning on IBM Easy Tier® space reservation

Available options to release allocated extents

The current implementation of the DS8900F thin provisioning does not provide a mechanism to release space on less than an entire logical volume. The following options are available to release allocated extents:

- ▶ Deleting an ESE volume releases all its allocated extents (real and virtual capacity). The real and virtual extents become available for reallocation.
- ▶ Requesting volume initialization through DS CLI or DS GUI releases all its real extents.
- ▶ A SCSI **Format Unit** command that is issued by the attached host releases its real extents.

Important: For critical applications, standard logical volumes might be a better choice to avoid any risk of running into an out-of-space situation.

2.2.5 Sizing

Small extents increase the number of total extents and the amount of system memory that is needed for metadata. Therefore, some restrictions apply in the current implementation. The size maximums that are listed in Table 2-1 compare the DS8000 physical and virtual volumes with small extents against physical and virtual volumes using large extents.

Table 2-1 DS8900F configuration limitations for large and small extents based on System Memory

Total System memory	Maximum physical size with large extents	Maximum virtual size with large extents	Maximum physical size with small extents	Maximum virtual size with small extents
≤ 512 GB	2 PB	4 PB	512 TB	1 PB
≥ 1 TB	8 PB	16 PB	2 PB	4 PB

The cache size is an important planning aspect and must be chosen when some metadata is kept in cache. Volume metadata is in the extent pool with the volumes. In hybrid extent pools with different tiers, metadata is placed in the highest, fastest tier. The preferred metadata location is in Flash or solid-state drives (SSDs). The amount of metadata might be large if the overprovisioning ratio is large. With hybrid DS8000s, metadata is not allowed to use all of the Flash or SSD space; only a portion of the Flash or SSD space is used.

Tip (for a hybrid DS8880): If your flash capacity is approximately 10% (or more) of the total capacity, all metadata is most likely in the flash tier.

2.2.6 Performance

You might think that thin-provisioned volumes have lower performance than standard volumes because some overhead is always involved in managing thin-provisioned volumes.

With older DS8000s and for good performance, a flash tier should be available for the user data and particularly to hold the volume metadata.

Performance measurements show nearly no performance effect that is associated with the DS8900F ESE implementation that uses large extents for thin provisioning and only a small overhead for small extents. The overhead mainly appears when new extents must be allocated by the system. Storage allocation is fast because of the fast initialization process, which is referred to as *Quick Init*, for newly allocated extents.

Defining new volumes requires slightly more time with the metadata structures that are introduced with the DS8900F compared to previous implementations. For extent pools with large extents, you can speed up the new volume creation process to the usual speed by pre-allocating some space for metadata.

By that use the `mksestg -vircap capacity pool_ID` command, metadata is pre-allocated for the specified virtual capacity in extent pool `pool_ID`. This virtual capacity is the sum of capacity of all the volumes that you want to define. This configuration might be useful for IBM z/OS environments in which it is common to define thousands of volumes in one step.

Note: Pre-allocating virtual capacity is not supported for extent pools with small extents. For more information, see Example 4-10 on page 39.

When sequential writes are performed to allocated storage, small extents can use many more disk drives in parallel as large extents. With large extents, the first 1 GiB writes are performed to one array until we skip to the next array. With small extents, we skip to the next array after 16 MiB are written. This way, more drives are active for this write stream. This configuration is true for fully- or thin-provisioned volumes (which is also true for CKD writes). From a performance perspective, ESE volumes often are suited for production volumes.

2.2.7 Applications

The efficiency of thin provisioning depends on the behavior of the applications, especially in the way that they allocate space for their data. You must carefully select the applications that can benefit from it. The application that performs this function often is a file system, or a database in cases where a file system is not used.

Ideally, the space allocation of the applications must have the following characteristics to benefit from thin provisioning:

- ▶ The application tends to localize its data in contiguous regions rather than scatter the data across the assigned ESE volume (such as IBM i).
- ▶ The application tends to reuse previously used space before it uses new space that was never used before. The use of new space might trigger the allocation of more real capacity extents to the assigned ESE volume. When the file system or database deletes data, the space that is used by this deleted data is not automatically released.
- ▶ You want the application storage needs to increase consistently over time.

The capacity allocation granularity of a specific implementation (1 GB /1113 cylinders (CKD) ESE versus 16 MiB /21 cylinders (CKD) ESE) might not be appropriate for certain applications.

The application (file system or database) might manage data in such a way that it ends up fully allocating the volume, even when the stored data at any certain time is much less than the full volume capacity.

The use of a Logical Volume Manager (LVM) is not expected to have a negative effect on thin provisioning. If the LVM allows a logical volume to be subdivided into partitions, it is advisable to create the partition boundaries that are aligned with an extent boundary of the logical volume to improve the likelihood that the extent remains deallocated. This need might not be easy to determine in practice.

If a file system is used, this file system is expected to be the determining factor regarding the interaction with thin provisioning. Databases often can operate with or without a file system. Without a file system, the database behavior is expected to be the determining factor regarding the interaction with thin provisioning. For example, if the database features the characteristic of initializing all of its configured table spaces, any associated ESE volumes are fully allocated, which makes thin provisioning useless in this case.

Note: It is beyond the scope of this paper to identify software or applications in relation to their compatibility with the DS8900F Thin Provisioning feature.

2.3 Planning for ESE volumes as FlashCopy targets

Thin-provisioned ESE volumes (particularly ESE volumes with small extents) are well-suited as IBM FlashCopy® target volumes. When a FlashCopy operation is performed, space that was in use at the target volume (space that was allocated on real storage) is released. All pointers indicate the original source volume. As changes occur on the source volume, extent allocations at the target volume occur and the original data blocks are copied to the target volume's extents. When a FlashCopy relationship is withdrawn, the target volume's space is released.

Note: For CKD volumes, space release is disabled in the case of a Fast Reverse Restore for Global Mirror journal FlashCopy.

A small extent size for the target volume is more capacity efficient than large extents. Therefore, you should consider thin-provisioned volumes with small extents as FlashCopy targets. You must decide whether you want to create these volumes in a separate extent pool or in the same extent pool as the production volumes.

FlashCopy is permitted between any type of volume, full or thin-provisioned.

Restriction: In the current implementation, a *preserve mirror* FlashCopy (also known as *Remote Pair FlashCopy*) does not release space on the secondary FlashCopy target volume.

2.3.1 Thin-provisioned ESE volumes for mirroring

Metro Mirror and Global Copy relationships can be between like volume types: full-provisioned to full-provisioned and thin-provisioned to thin-provisioned. They might also be between thin-provisioned and full-provisioned and vice-versa. However, in the case of a full-provisioned volume being mirrored to a thin-provisioned volume, it becomes full provisioned.

The extent size does not matter for ESE volumes; however, you must allow the target volume to be space efficient (for example with the DS CLI `mkpprc` option `-tgtse`).

Mirror establishes releases space on the target volume and allocates space according to source volume attributes.

Global Mirror

The best use case for thin-provisioned volumes (particularly with small extents) is the Global Mirror journal FlashCopy target volumes at the secondary site.

However, releasing space is a time-consuming operation. Global Mirror performs a FlashCopy to form consistency groups as frequently as every 2...5 seconds. Therefore, space is not released on every FlashCopy operation. Instead, the DS8900F decides on a load basis when to perform the next release of space.

With 16 MiB or 21 cylinder (CKD) extents and no immediate space release, the capacity that is allocated for FlashCopy target volumes is higher than 1%. Planning for approximately 20% to 30% capacity for FlashCopy targets is recommended.

Important: For IBM i, the following items need to be considered.

- ▶ For thin provisioned FlashCopy targets, the recommendation is 50%.
- ▶ Global Mirror journal volumes can be thin-provisioned and recommended to be sized at 50%.

z/OS Global Mirror

A z/OS Global Mirror source volume can be thin-provisioned. You can define a thin-provisioned ESE volume as secondary volume, but it will become fully provisioned.

2.3.2 Choosing an extent size

Part of the planning process is to choose an extent size for each extent pool.

The recommendation is to use small extents when most of your volumes in an extent pool are thin-provisioned unless you have many performance needs that must be met. If you do not use thin provisioning and in case the machine holds a huge capacity, use the large extent size.

When an extent size is chosen, it cannot be changed for the extent pool. You can take out a rank from an extent pool with the depopulation function if there is space that is left on the other ranks in this extent pool to which to move the data. Then, you can delete the rank and re-create it with another extent size and add it to an extent pool with the corresponding extent size.

2.3.3 Setting up monitoring for capacity utilization

Part of the planning should include what to do when free space becomes low and how to detect it. The DS8900F provides means to reserve capacity and to set warning thresholds when free capacity gets low.

For more information, see Chapter 5, “Managing and monitoring thin-provisioned volumes” on page 47.

2.3.4 ESESizer tool

In case the space estimations in “Thin-provisioned ESE volumes for mirroring” are not precise enough, a tool is available for help. With recent IBM Copy Services Manager versions like 6.3, a CSM ESESizer session is available for DS8900F and DS8880 that can regularly query a write-monitoring bitmap of the DS8000. The output of query can be used to calculate how much capacity is allocated over time for both functions of either ESE FlashCopy, or Safeguarded Copy.

For more information about how to use the ESESizer tool, see:
<https://www.ibm.com/support/pages/node/6372180>



Space-efficient volumes in the IBM DS8000

This chapter describes the design and implementation of Extent Space Efficient (ESE) volumes in the DS8000. As described in Chapter 1, “Thin provisioning” on page 1, ESE volumes can be created with small or large extent sizes.

Thin provisioning with ESE volumes is available with the basic license for Fixed Block (FB) and Count Key Data (CKD) volumes.

Applications and operating systems that are appropriate for thin provisioning can use ESE logical volumes for their storage needs.

A repository is optional for ESE volumes. For more information, see 3.1.1, “Optional ESE repository for thin-provisioned logical volumes” on page 20.

3.1 DS8000 logical volumes

The following types of volumes are available with DS8900F:

- ▶ Standard logical volumes

These volumes are the regular type of volumes and are fully provisioned with real capacity.

- ▶ ESE volumes with large extents

These volumes are implemented to allow thin provisioning and are dynamically provisioned by using the DS8900F regular extents. As with the first DS8000 models, the extent size can still be 1 GiB for FB volumes and 1113 cylinders for CKD volumes. This extent size is referred to as *large extents*.

When an ESE logical volume is created, the volume has no real data capacity. However, the DS8900F uses some real data capacity for metadata that it uses to manage space allocation. The metadata holds information about extents and volume blocks that are allocated in a rank. This metadata that is used for thin provisioning allows the DS8900F to determine whether new extents must be allocated.

Tip: Use extent pools with large extents if you want to use fully provisioned volumes and do not plan to use thin-provisioned volumes.

- ▶ Extent space-efficient volumes with small extents

Thin provisioning is much more capacity efficient with a smaller extent size. Particularly, when data is deleted, the probability that a small extent is empty and can be released is much higher than with large extents. In previous implementations, few methods were available to reclaim the space that was deleted by a host (for example when you delete a data set).

For storage pools with large extents, metadata is also allocated as large extents (1 GiB for FB pools or 1113 cylinder for CKD pools). Large extents that are allocated for metadata are subdivided into 16 MiB subextents for FB volumes, or 21 cylinders for CKD volumes. These subextents are referred to as *metadata extents*.

For extent pools with small extents, metadata extents are also small extents. A total of 64 metadata subextents are in each large metadata extent for FB, and 53 metadata subextents are in each large metadata extent for CKD.

For each FB volume that is allocated, an initial 16 MiB metadata subextent or metadata small extent is allocated. Another 16 MiB metadata subextent or metadata small extent is internally allocated for every 10 GiB of allocated capacity or portion of allocated capacity.

For each CKD volume that is allocated, an initial 21 cylinders metadata subextent or metadata small extent is allocated. Another 21 cylinders metadata subextent or metadata small extent is internally allocated for every 11130 cylinders (or 10 3390 Mod-1) of allocated capacity or portion of allocated capacity.

For example, a 3390-3 (that is, 3339 cylinders or about 3 GB) or 3390-9 (that is, 10,017 cylinders or 10 GB) volume takes two metadata extents (one metadata extent for the volume and another metadata extent for any portion of the first 10 GB). A 128 GB FB volume takes 14 metadata extents (one metadata extent for the volume and another 13 metadata extents to account for the 128 GB).

Metadata extents with free space can be used for metadata by any volume in the extent pool.

Important: In a multi-tier extent pool, volume metadata is created in the highest available tier. In a hybrid DS8880, it is recommended to include some flash or solid-state drive (SSD) storage for the metadata.

With this concept of metadata extents and user extents within an extent pool, some planning and calculations are required. For example, planning and calculations are required in a mainframe environment in which thousands of volumes are often defined and the entire capacity is allocated during the initial configuration. You must calculate the capacity that is used by the metadata to determine the capacity that can be used for user data. This information is only important when full provisioned volumes are used. Thin-provisioned volumes use only metadata when created and space is used over time when data is written.

For extent pools with small extents, the number of available user data extents can be estimated by using the following calculation:

$$(\text{user extents}) = (\text{pool extents}) - (\text{number of volumes}) - (\text{total virtual capacity})/10$$

When the details of the volume configuration are not known for large extent pools, it is possible to estimate the number of metadata extents based on averages of many volumes only. The following calculations are used:

- ▶ FB Pool Overhead = (Number of volumes x 2 + total volume extents/10)/64 and rounded up to the nearest integer
- ▶ CKD Pool Overhead = (Number of volumes x 2 + total volume extents/10)/53 and rounded up to the nearest integer

These formulas overestimate the space that is used by the metadata by a small amount because they assume waste on every volume; however, the precise size of each volume does not need to be known.

Consider the following examples:

- ▶ A CKD storage pool has 6,190 extents in which you expect to allocate all of the space on 700 volumes. Your Pool Overhead is 39 extents by using the following calculation:
$$(700 \times 2 + 6190/10)/53 = 38.09$$
- ▶ An FB storage pool has 6,190 extents in which you expect to use thin provisioning and allocate up to 12,380 extents (2:1 overprovisioning) on 100 volumes. Your Pool Overhead is 23 extents by using the following calculation:
$$(100 \times 2 + 12380/10)/64 = 22.46$$

To confirm sizing requirements, see 2.2.5, “Sizing” on page 12 and Table 2-1, “DS8900F configuration limitations for large and small extents based on System Memory” on page 12.

Space for a thin-provisioned volume is allocated when a write occurs. More precisely, it is allocated when a destage from the cache occurs and insufficient free space is left on the currently allocated extent.

Therefore, thin provisioning allows a volume to exist that is larger than the physical space in the extent pool to which it belongs. This approach allows the “host” to work with the volume at its defined capacity, even though insufficient physical space might exist to fill the volume with data.

The assumption is that the volume will never be filled, or as the DS8000 runs low on physical capacity, more capacity will be added. This approach also assumes that the DS8000 is not at its maximum capacity.

Thin-provisioned volume support is contained in the Base Function license group. Thin provisioning is an attribute that you specify when a volume is created.

The Quick Initialization process

Allocating usable space for a DS8000 (standard) volume occurs when the volume is created. The volume creation is done from the DS CLI or the DS GUI. A process starts in the background to format the entire volume. Although the formatting process is on-going, the volume is immediately accessible for servers.

The Quick Initialization process significantly shortens the time that is required to format a volume. In addition, Quick Initialization enables all copy services for standard logical volumes to be established immediately. The Quick Initialization process provides the capability to allocate individual extents dynamically (when required) by a write operation to an ESE logical volume.

3.1.1 Optional ESE repository for thin-provisioned logical volumes

A repository is a protected reserve for extents. An ESE repository is not needed, but you might want to consider it because it provides some advantages.

Fully-provisioned and space-efficient volumes can coexist in an extent pool, as shown in Figure 3-1. Although fully-provisioned volume capacity is ensured, the same is not true for thin-provisioned volumes.

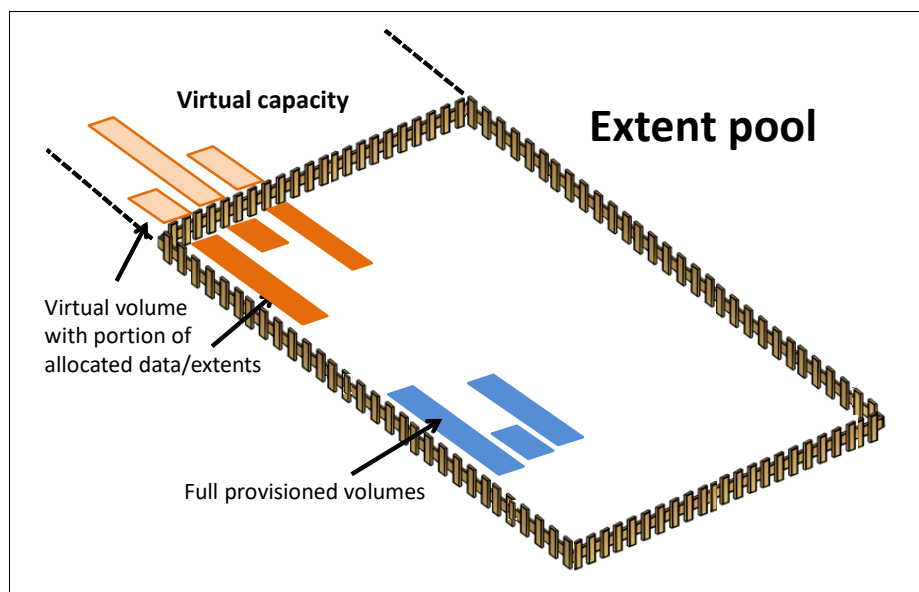


Figure 3-1 Fully-provisioned and thin-provisioned volumes in an extent pool

Reserving storage

A storage administrator might think that plenty of free capacity is available and can allocate all free capacity to fully provisioned volumes. The next write that requires a new extent allocation for a thin-provisioned volume is no longer possible because no free extent is available. Soon, all of your thin-provisioned volumes have this problem and I/Os to these volumes stops.

To continue, some space must be freed or (physical) capacity added. To avoid such a situation, it is possible to reserve some capacity from the beginning if you work with thin-provisioned volumes in pools with large extents. You can reserve some capacity by enabling the extent-limit function for an extent pool by using the **chextpool -extentlimit enable -limit 90 pool_ID** command. The use of this command shrinks the extent pool to 90% of its size and reserves 10% for future use, as shown in Figure 3-2.

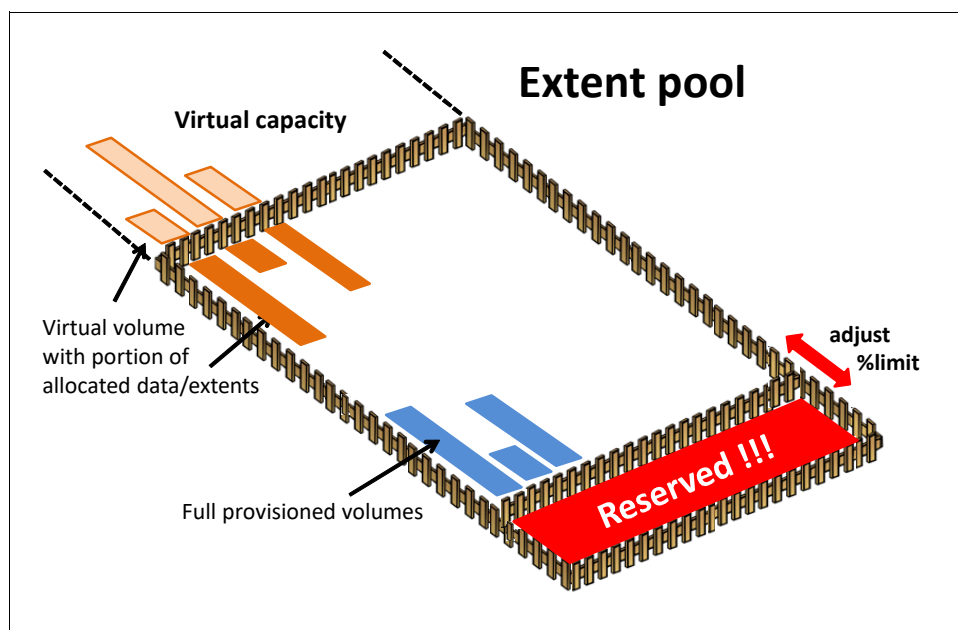


Figure 3-2 Extent pool with a limit to reserve storage

When required, you can change the limit and set it to 95% or 100%, for example. This change frees space within the extent pool.

However, the problem still exists that the capacity for thin-provisioned volumes can be taken, which leaves no space for new extent allocations when fully provisioned volumes are allocated. To prevent this situation, you can create a *Repository*, which is described next.

Repository

By using the DS CLI `mksestg -repcap capacity pool_ID` command, capacity is reserved for extent space efficient volumes and all extents from thin-provisioned volumes are placed in this fenced area. See Figure 3-3.

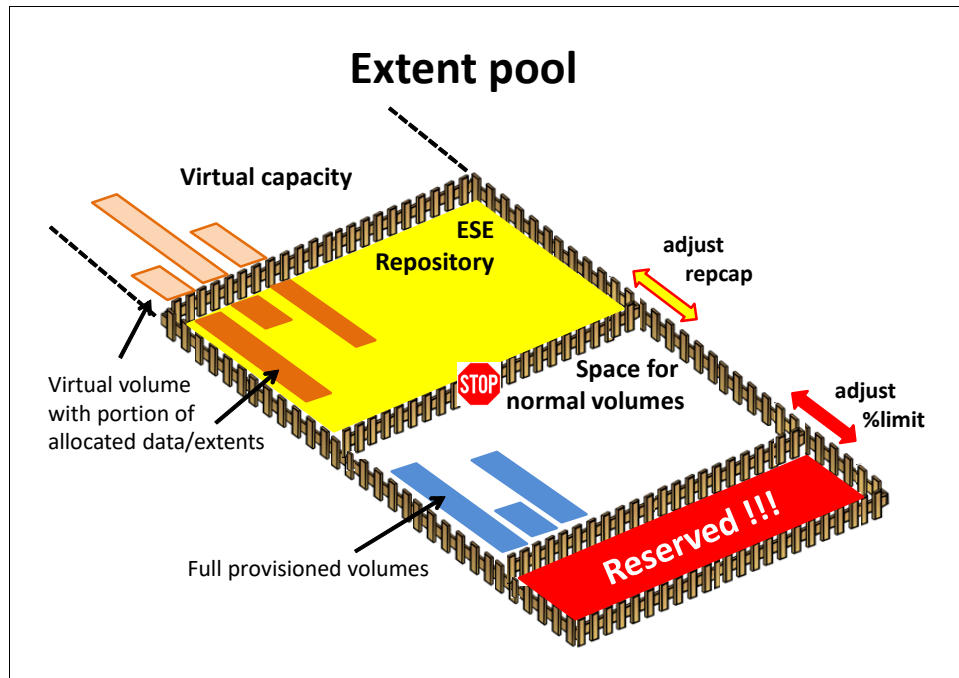


Figure 3-3 Extent pool with a repository for extent space efficient volumes

Virtual capacity

You can also specify a *virtual capacity* by using the `mksestg` command or by using the `chsestg -viricap vircapacity pool_ID` command if you created a repository.

Restriction: Pre-allocating virtual capacity applies to extent pools with large extents only. For more information, see Example 4-10 on page 39.

However, this specification is possible for an extent pool with *large* extents only. You do *not* limit the virtual capacity by using this command. Instead, you pre-allocate metadata extents for the specified virtual capacity. The use of this command can speed up the volume creation process, particularly when thousands of volumes are created as it is common for z/OS environments.

Note: It is not possible to set the repcap and viricap controls by using the DS GUI.

An extent pool can contain a maximum of one ESE repository. The ESE repository can be created before or after the ESE logical volumes are created. If created after, any ESE logical volumes are automatically added to the repository. In this situation, the size of the repository must be large enough to contain all of the allocated extents that are being used by the ESE logical volumes.

The size of the repository can be set up to the total available space in the extent pool, but it is still included in the available space when the available space is displayed in the extent pool. Although it is included in the available space, it cannot be used for anything but storing data on ESE logical volumes.

The size of the ESE repository can be changed dynamically to increase or decrease its size, and delete it altogether.

ESE volumes without an ESE repository

When ESE logical volumes without a repository are used, no reserved space is available for storing data on those volumes. The amount of space that is available to write data on the ESE logical volumes is the available space in the extent pool. If the available space is reduced because new standard or ESE logical volumes are created, the amount of space that is available to write data on the current ESE logical volumes is also reduced because reserved space does not exist for the ESE logical volumes.

When an extent pool is created, the virtual capacity of the pool is zero. When an ESE logical volume is created, the virtual capacity is automatically increased to support the size of the ESE volume. The virtual capacity of an extent pool is limited. This limit is based on using all of the available space in the extent pool for metadata for space-efficient volumes. Because ESE logical volumes do not require a repository, all of the extent pool can be used for metadata, but no space exists for writing data to the ESE logical volumes.

Therefore, it is important to plan the space for the ESE logical volumes so that adequate space exists in the extent pool to store all the planned data that might be written to volumes and to accommodate the virtual capacity of the volumes. The **opratio** parameter is the difference between the virtual capacity of the volumes and the space on which to store data on the volumes.

The following options are available to plan out the ESE logical volumes:

- Use an existing extent pool:
 - a. Determine the physical space (or available space) in the extent pool.
 - b. From that number, subtract the amount of space you require to store data on all the ESE volumes that you plan to create in the extent pool. The remaining capacity is available for ESE volume metadata.
 - c. Multiply that number by 200 to calculate an approximate virtual capacity for the extent pool that can be used to create ESE logical volumes. The problem that occurs is that the **opratio** likely is high and not at all practical. This calculation that uses real numbers is shown in Example 3-1.

Example 3-1 Existing pool

Available capacity (GUI) 481,694 GiB

Total extent pool capacity 2,122 GiB - actual physical capacity

Space for ESE volume data 1,000 GiB - your decision

Allocated extent pool space 0 GiB - assumes no standard volumes

Space available for metadata .. 1,122 GiB

Approximate total virtual capacity for ESE volumes is 224,400 GiB

This would result in an **opratio** of 224:1 - an high value

- Determine the **opratio** that you want, with as low a number as possible to start. You might be able to increase this number if usage allows it, as monitored over time. Multiply the **opratio** by the space that you want to be used to store data for the ESE volumes that are created. The result is your total virtual capacity in the extent pool to create ESE volumes. This calculation that uses real numbers is shown in Example 3-2 on page 24.

Example 3-2 Using opratio

Available capacity (GUI) 481,694 GiB

Total extent pool capacity 2,122 GiB - actual physical capacity
Space for ESE volume data 2,080 GiB - your decision
Allocated extent pool space 0 GiB - assumes there is standard volumes
Space used for metadata 42 GiB
Opratio 4:1

Approximate total virtual capacity for ESE volumes is 8,320 GiB

- ▶ A more practical approach is the following process:
 - a. Determine your ESE volume capacity requirements.
 - b. Divide that number by your opratio requirement.
 - c. Add space for metadata (ESE volume capacity multiplied by 0.5).

This calculation provides you with the required storage capacity when you set up the extent pool. This calculation that uses real numbers is shown in Example 3-3.

Example 3-3 ESE capacity requirements

Total ESE volume capacity 10,000 GiB Opratio required 4:1
Space for ESE volume data 2,500 GiB
Space required for metadata 50 GiB

Total extent pool capacity 2,550 GiB

Tools are available to assist you in planning your overall storage requirements in a DS8900F. For more information, see Chapter 2, “Planning considerations and guidelines” on page 7.

The DS8900F creates the virtual capacity that is needed for ESE logical volumes as they are created. Without an ESE repository, the real capacity is allocated by using free extents from the extent pool. Extents for standard logical volumes (if they exist) and dynamically allocated extents for ESE logical volumes are allocated independently from the same extent pool. When the extent pool has more than one rank, the dynamic allocation of extents follows the usual extent allocation methods. That is, rotate extents or rotate volumes, depending on what was specified when the ESE logical volume was created.

ESE volumes with an ESE repository

The repository for ESE volumes can be created before any ESE volumes are created, or after the volumes are created.

Consider the following points when an ESE repository is defined:

- ▶ The actual allocated size is rounded up to the next highest whole number percentage of the current physical extent pool capacity, not what is specified.
- ▶ You can dynamically increase or decrease the size of the repository. If storage is allocated in the repository, the size cannot be lower than the current allocation.
- ▶ You can specify a size that starts at 0 GB.
- ▶ If created after the ESE volumes exist, those volumes and any other volumes use the repository space for storing data on the volumes.
- ▶ The ESE repository can be deleted without deleting any volumes, even if data is allocated.
- ▶ No overhead is required for the repository.

Note: When you choose the capacity of the ESE repository, it can be dynamically changed to meet future needs, as required. If the repository is created before the ESE volumes are created, space must be available in the extent pool for the metadata of the volumes that you plan to create.

When you work with extent pools, an available space is indicated that is based on the current configuration. Unfortunately, when you create an ESE repository, the space that is designated for the repository remains in the available space for the extent pool because the repository can be removed at any time. Only as space in the repository is used to store volume data is the available space reduced by the amount of the repository space that is used.

Remaining space in the repository is still considered available in the extent pool. Although it is indicated as available space in the repository, the space cannot be used for anything else but storing data for ESE logical volumes. Therefore, you might encounter an out-of-space condition when a fully provisioned volume is allocated.

To calculate the real available capacity for normal, fully provisioned volumes, you must perform the following tasks after the ESE repository is created (see Figure 3-4):

1. Determine the available capacity by using the `lsextpool` DS CLI command.
The number that is shown are the extents. For extent pools with small extents, you must calculate the GiB by multiplying the number of extents with the extent size.
2. Subtract the ESE repository capacity by using the `lssestg` DS CLI command.
3. Add the ESE repository allocated capacity by using the `lssestg -l` DS CLI command.

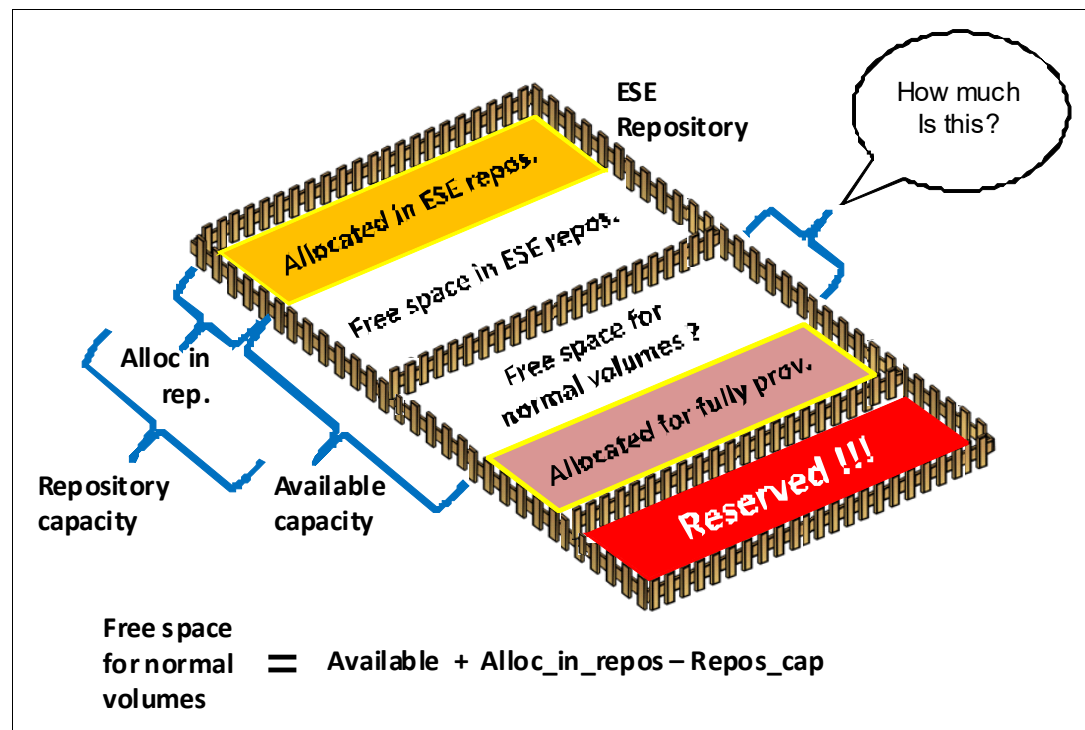


Figure 3-4 Calculation of free capacity for normal volumes

3.1.2 Capacity allocation for ESE volumes

When a host writes data to an extent of an ESE logical volume and the extent does not have an associated real extent, the write operation is held in abeyance until a real extent from the extent pool is dynamically allocated to the ESE logical volume. If a repository is used, an available extent that belongs to the repository is allocated. If no repository is used, an available extent in the extent pool is allocated. This process is applied primarily in the cache of the DS8000 storage facility image.

The write data is accepted into the write cache. Whenever a real extent is allocated to an ESE volume, a quick initialization is started to dynamically initialize the tracks on the real extent that were not written by the requesting write operation. This quick initialization is performed in parallel with the host access. When the quick initialization finishes, the extent is ready for normal operations. Any other write operations that might cause one or more destages of tracks can now also be applied to the new extent.

Depending on the incoming I/O operation and the status of the volume, the following situations can occur and are handled:

- ▶ Write access to a DS8000 block within the allocated capacity
The block is allocated in an extent. The write operation continues normally, where data is stored in the write cache and the write is committed to the application server. Upon a destage from cache to disk, data is written to the appropriate block in the extent that was allocated.
- ▶ Write access to a DS8000 block where no capacity is allocated
The DS8900F dynamically allocates to the logical volume the capacity that is needed to store the write-data to disk. For an ESE logical volume, the allocated capacity is one extent. The data is stored into the write cache and the write is committed to the application server. Then, unused data in the extent is initialized (by using Quick Initialization). Upon a destage from cache to disk, data is written to the appropriate block in the extent that was allocated.
- ▶ Read access to a DS8000 block with allocated capacity
This situation is the same situation as in a standard logical volume. When the data is not in the read cache, a stage operation directs the read operation to the logical block and loads the data into the read cache for the next potential reads.
- ▶ Read access to a DS8000 block where no capacity is allocated
The data for this logical block is synthesized in the read cache with an initialization pattern. Read operations to unallocated capacity normally do not occur from an application perspective because applications often read data that is stored on disk. That is, it is unlikely that data is used when it was never written. From the perspective of the operating systems and lower levels, reads from deallocated capacity can occur because all blocks of volume (including the virtual capacity) can be addressed by the host.

The extent pool is equipped with parameters to allow monitoring of the real capacity utilization. The most important parameters are listed. These parameters are available for real and for virtual capacity. The following values can be obtained for each extent pool by using the DS CLI **showextpool** command:

- ▶ configured
Gives the number of real extents that are configured when pool.
- ▶ allowed
The number of real extents that remain when a limit is set. If the limit is disabled or is set to 100%, the allowed extent has the same number as the configured extent.

- available

The number of extents that are free to be allocated by standard logical volumes or ESE logical volumes. If an ESE repository exists, available repository extents are included in this number.

- allocated

The value of allocated real extents. It is the sum of extents that are allocated in standard logical volumes and ESE logical volumes.

3.1.3 Out-of-space condition

The write operation is rejected by the DS8900F when a write operation requires another extent allocation but all real extents are used.

If no ESE repository is used, this out-of-space condition remains until new physical capacity is made available to the DS8000 storage facility image. Therefore, one or more ranks must be assigned to the specific extent pool. If ranks are available in the DS8000 frames but are not yet configured, they can be configured to the extent pool.

If ranks were configured to the extent pool as reserved ranks, more capacity is made available by unreserving one or more of those ranks. Otherwise, one or more new ranks must be physically installed and then added to the extent pool. If you reserved space in the extent pool with the extent-limit function, you can make the reserved space smaller.

If an ESE repository is used, you can manage the situation by using one of the following methods:

- If free extents are used in the extent pool, increase the size of the repository to use up to all of the free extents. This increase alleviates the problem, but it might be temporary only, depending on future allocation requirements and how much space was added to the repository.
- If no free extents are in the extent pool, physical capacity must be added to the extent pool. When the physical capacity is added, you can increase the repository size by using the **chsestg -repcap** command, which alleviates the out-of-space condition.

Important: Out-of-space conditions in an extent pool must be avoided in any case because this issue results in application access loss.

An extent pool is characterized by parameters and functionality to allow monitoring of its capacity.

One parameter is the extent threshold for the extent pool. This parameter is important when operating without an ESE repository. The threshold is set as a percentage of the number of remaining available extents. For example, if the threshold is set to 15% (which is the default), a Simple Network Management Protocol (SNMP) trap is sent out to the data center management console as a warning that the remaining available capacity for this extent pool is below 15% of the overall initial capacity. However, real capacity (extents) can still be allocated.

An extent limit can be configured for extra safety. This limit provides a restricted amount of capacity that is not initially accounted for as free extents in the extent pool. This limit also gives you another reserve that can be brought into the extent pool if you cannot install more capacity quickly enough after you receive the first warning from the extent threshold that the available capacity is running low.

When you use an ESE repository, you can use repository parameters to monitor the allocation of extents in the repository. One parameter is the **repcapthreshold**, which is used to set the minimum percentage of available space in the repository. If the percentage of available space falls below this parameter, the repository is in a status of threshold exceeded.

This parameter is one of three threshold settings for a repository and is referred to as the “user threshold”. The other two thresholds of 0% (repository full) and 15% (85% full) are system controlled and cannot be changed. (For more information about this parameter, see the **mksestg** command in the [DS CLI commands documentation](#)). After a threshold is met, a notification is sent (if notifications are configured by using the **chsp** command).

The layout of the extent pool when an ESE repository is used is shown in Figure 3-5.

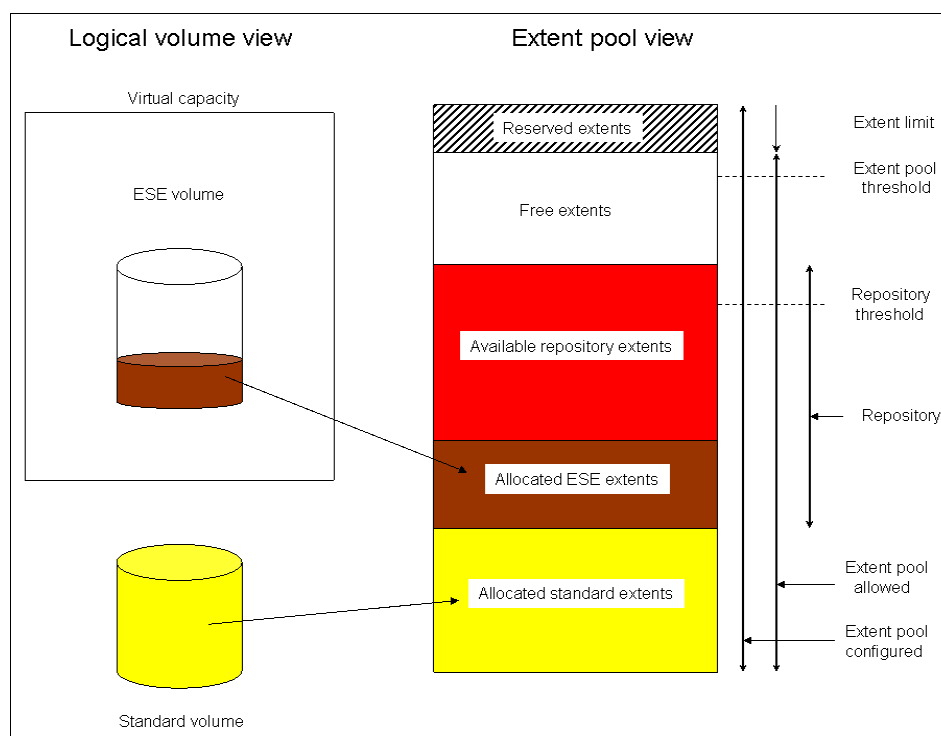


Figure 3-5 Extent pool space with an ESE repository

3.1.4 Volume creation

ESE logical volumes are created by using virtual extents. The ESE logical volume initially includes some real capacity that is allocated. This capacity is necessary to store the volume’s internal metadata and is referred to as *overhead*. The metadata consists of a structure and data about the extent, such as status information, address information, and other information that is used by the DS8900F microcode. Although the capacity that is required by metadata is low (less than 0.5% of the logical volume capacity), it uses real storage. The overhead per extent pool can be shown by using the DS CLI **showsestg** command.

The more volume capacity that is created in the pool, the more metadata space is required, which reduces the amount of space to store data on the volumes. Without a repository setup for the ESE logical volumes, the available space to store data on the ESE volumes decreases as you add more ESE or standard volumes.

When an ESE logical volume is made accessible to the host, certain allocation operations might be executed by the host to make the volume usable by the operating system and the applications. Depending on the host platform, the allocation tasks differ. For example, on an IBM AIX® host, the volume is allocated to the volume manager by creating a volume group first. During this process, the Volume Group Descriptor Area (VGDA) and Volume Group Status Area (VGSA) are written to the volume, which causes the dynamic allocation of an extent.

Allocation of the logical volume on the AIX host also causes some writes to the volume. The Enhanced Journaled File System (JFS2) then causes another write operation. These steps occur at the AIX host with JFS2 result in two extents being allocated at the DS8000 back end. For more information, see Chapter 6, “Use of thin-provisioned volumes in IBM Z” on page 53.

Operating system platforms or applications that cause the storage subsystem to claim the whole volume capacity when attaching the DS8900F logical volume cannot benefit from thin provisioning. Such operating systems or applications must use standard logical volumes.

3.1.5 Releasing space

After a file system or application allocates its initial storage and is turned over to production, new data eventually is added, which results in new storage allocation. Data also is modified or deleted. These operations produce portions of unused space on the volume from the host perspective and on the logical volumes of the storage subsystem. This situation is known as *fragmentation* and occurs on space-efficient logical volumes and on standard logical volumes in the same way.

For standard logical volumes, this situation is almost insignificant because the entire capacity (used or unused by the application) is exclusively assigned to the logical volume. It is more important to ESE logical volumes because unused capacity that results from data deletions cannot be reused for the thin provisioning until it is released back to the real capacity pool.

The method to release space back to the real capacity pool is to remove the ESE logical volume or to issue an `initfbvol` or `initckdvol` command, which initializes the entire volume.

With z/OS 2.1 and later, a function is available that allows the physical space release. For more information, see Chapter 6, “Use of thin-provisioned volumes in IBM Z” on page 53.

In Open Systems environments, you can perform a space release by using the Veritas Volume Manager (Veritas InfoScale). For more information about hardware compatibility for your respective OS and InfoScale version regarding DS8000, see the [Veritas Services and Operations Readiness Tools \(SORT\) page](#) of the Veritas website.

For more information about releasing space with Veritas, see the Storage Foundation Administrator's Guide for the respective operating systems.

Generally, FB hosts can issue the UNMAP or Write Same with UNMAP commands to release space for a certain logical block address (LBA) range. The DS8000 releases space internally, as soon as full extent is covered. For instance, Linux users can use the `fstrim` and/or `sg_unmap` commands. Other host operating systems might have specific commands. For more information, see the documentation for your operating system.

With IBM i, you cannot currently perform a space release; with VMware hosts, the space release is restricted to simplex volumes.

3.1.6 The DS8900F flash backend

Starting with code releases R9.1 and later, additional improvements are introduced to make the DS8000 handling of thin volumes more efficient in its flash drive backend. Freed, unallocated, or initialized space on RAID arrays is zeroed with the **Write Same with UNMAP** command. This command is issued to the flash drive to free the pages in the flash drive. The pages then go back to the free pool of capacity within the drive. This saves additional Program/Erase cycles when extents are freed and reallocated, and thereby enhances the endurance of the flash drives. Also, internal drive performance improves, in areas such as flash drive garbage collection and the free-page management for new writes.

Support is also added to not rebuild a RAID stripe if all space is not mapped at the flash drive level, namely if a parity stripe is completely unmapped. In this case, in a rebuild, this support prevents additional writes and P/E cycles to the target drive during that rebuild. Also, the rebuild for partially-allocated RAID arrays will be carried out faster.

3.1.7 Migration considerations

Considerations are described in this section about how to bring thin provisioning into production. It is assumed that the applications and the operating system on which the applications are running are viable candidates for thin provisioning.

Applications that currently use standard logical volumes must be carefully analyzed to determine how much storage capacity is not used from the server perspective. ESE logical volumes are then assigned to the server and data can be migrated from the original standard volumes to ESE volumes.

At the end of the migration, the real capacity that is allocated to the ESE logical volume must be equal to the capacity that was used on the standard logical volume. Depending on the application and operating system, more capacity might be allocated. This need might occur because of the extra metadata that is required for thin provisioning and the behavior of the file system.

The actual data must be migrated by using the host-based application methods or a Logical Volume Manager (LVM) based mirroring. Other methods that are provided by lower levels of the operating system might not produce the intended result. For example, the use of the UNIX **dd** command copies each block of the source device to the target device, which allocates all of the capacity on the ESE logical volume.

3.1.8 Performance considerations

A trade-off often exists between space-efficiency and performance that depends on the granularity of capacity allocation. A small granularity capacity allocation leads to the best space efficiencies. However, it might have a small performance affect with capacity allocation and the overhead that is involved with accessing metadata that determines whether the capacity is allocated.

When large extents are used in the DS8900F, the overhead is small. Therefore, the performance of I/O operations on such an extent must be nearly identical to that of a standard logical volume. The overhead is slightly higher for extent pools with small extents.

The time that it takes to access metadata also influences the overall performance. In a hybrid pool with some flash involved, a hybrid DS8880 allocates metadata in the highest pool in Flash or SSD, if possible.



DS GUI and DS CLI support for thin provisioning

This chapter describes the use of the DS GUI and DS CLI commands in support of thin provisioning-related functions and includes the following topics:

- ▶ 4.1, “DS GUI and DS CLI support to set up thin provisioning” on page 32
- ▶ 4.2, “Creating ESE thin-provisioned volumes” on page 42
- ▶ 4.3, “Summary” on page 46

4.1 DS GUI and DS CLI support to set up thin provisioning

When you create a volume, you must tell the system what type of volume you want to create. You have the choice between fully provisioned volumes and thin-provisioned Extent Space Efficient (ESE) volumes.

For the purposes of this IBM Redpaper publication, we are interested in the creation of ESE volumes. The size of the extent of the volume that you create is determined by the extent pool in which you create the volumes.

Creating extent pools is a standard task when a DS8000 is configured. Extent pools with large extents and extent pools with small extents can be used for thin-provisioned volumes. This section describes the creation of extent pools with small extents, which is more suited for thin provisioning. Therefore, we describe creating ranks and extent pools first.

4.1.1 Creating an extent pool with small extents by using the DS GUI

In the DS GUI, browse the Arrays by Pool window (see Figure 4-1) and click **Create pool pair**.

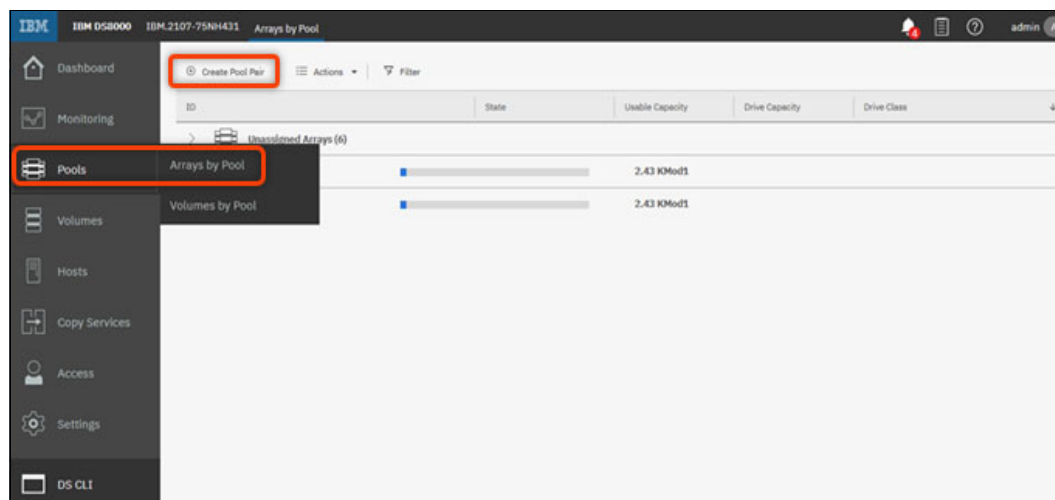


Figure 4-1 DS GUI panel to start with the creation of extent pools

Creating an extent pool and creating ranks are two different tasks. The first rank that is added to an extent pool determines the extent size for the entire pool. However, you cannot create an empty extent pool by using the DS GUI.

You must add at least one rank when an extent pool is created, which means that you must specify the extent size that you want to use in this extent pool.

Because we do not want to create a standard extent pool (with large extents), select the **Custom** pool creation method, as shown in Figure 4-2.

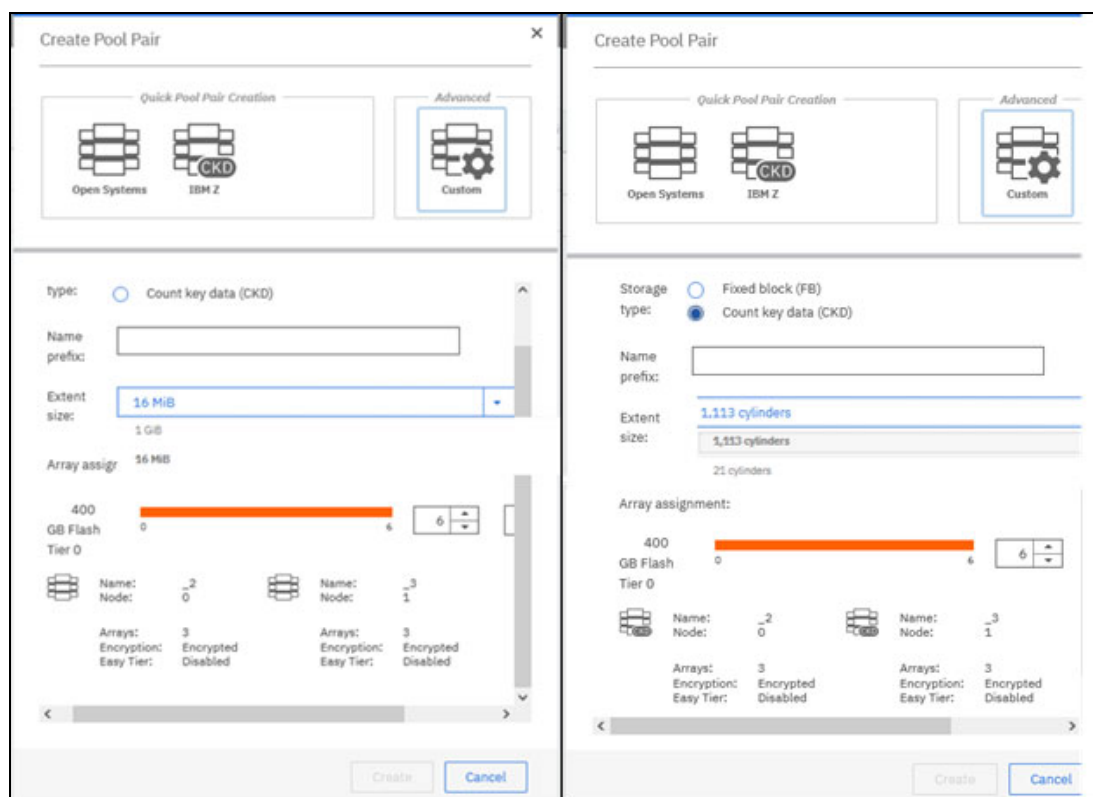


Figure 4-2 Selecting an extent size for an extent pool

Within the panel, you can select the extent size that is needed. If you choose to create an extent pool for FB volumes, you can select an extent size of 1 GiB or 16 MiB. If you want to create an extent pool for CKD volumes, you can choose between a 1113 cylinder and 21 cylinder extent size.

4.1.2 Creating an extent pool with small extents by using the DS CLI

By using the DS CLI, we can create an empty extent pool in which the extent size is not yet determined. As shown in Example 4-1, we created two extent pools: One for FB volumes and one for CKD volumes. We include the term *small* in the pool name; however, the extent size is still undefined.

Example 4-1 Creating empty extent pools

```
dsccli> mkextpool -rankgrp 0 -stgtype fb P4-FB-small
CMUC00000I mkextpool: Extent pool P4 successfully created.

dsccli> mkextpool -rankgrp 1 -stgtype ckd P5-CKD-small
CMUC00000I mkextpool: Extent pool P5 successfully created.
```

When we query the extent pool by using the **showextpool** command, we see that the pool is empty and does not include a defined extent size, as shown in Example 4-2 on page 34 for our FB extent pool.

Example 4-2 Status of an empty extent pool

```
dscli> showextpool p4
Name                P4-FB-small
ID                  P4
stgtype              fb
totlstor (2^30B)    0
availstor (2^30B)   0
resvdstor (2^30B)   0
rankgrp             0
numranks             0
numvols              0
status              full
%allocated           100
%available            0
configured           0
allowed              0
available            0
allocated            0
reserved             0
configuredCap(MiB/cyl) 0
allowedCap(MiB/cyl)  0
availableCap(MiB/cyl) 0
allocatedCap(MiB/cyl) 0
reservedCap(MiB/cyl) 0
%limit               100
%threshold           15
virextstatus         full
%virallocated        0
%viravailable        0
virconfigured        0
virallowed           0
viravailable         0
virallocated         0
virreserved          0
%virextlimit         -
%virextthreshold     -
encryptgrp           -
%allocated(ese)      -
%allocated(rep)      -
%allocated(std)      -
%allocated(over)     -
%virallocated(ese)   -
%virallocated(tse)   -
%virallocated(init)  -
%migrating(in)       -
%migrating(out)      -
numtiers             0
etmanaged            yes
etmigpauseremain     -
etmonpauseremain     -
etmonitorreset       unknown
extsize            -
```

The next step is to create one or more ranks and assign them to an extent pool. We can create and assign in one step, as shown in Example 4-3. Here, we created an FB and CKD rank and assigned them to the corresponding pools.

Example 4-3 Creating ranks with a certain extent size

```
dscli> mkrank -array A3 -stgtype fb -extpool P4 -extsize 16mib
CMUC00007I mkrank: Rank R0 successfully created.
dscli> mkrank -array A11 -stgtype ckd -extpool P5 -extsize 21cyl
CMUC00007I mkrank: Rank R1 successfully created.
```

When a **showextpool** query is run, we can see that the extent pool features assigned space and that the extent pool is a fixed extent size, as shown in Example 4-4 for the CKD pool.

Example 4-4 Extent pool status with ranks added

```
dscli> showextpool p5
Name                P5-CKD-small
ID                  P5
stgtype             ckd
totlstor (2^30B)    1532
availstor (2^30B)   1532
resvdstor (2^30B)   0
rankgrp             1
numranks            1
numvols             0
status              below
%allocated          0
%available          100
configured          92152
allowed             92152
available         92152
allocated           0
reserved            0
configuredCap(MiB/cyl) 1935192
allowedCap(MiB/cyl)  1935192
availableCap(MiB/cyl) 1935192
allocatedCap(MiB/cyl) 0
reservedCap(MiB/cyl) 0
%limit              100
%threshold          15
virextstatus        full
%virallocated        0
%viravailable        0
virconfigured        0
virallowed           0
viravailable         0
virallocated         0
virreserved         0
%virextlimit         -
%virextthreshold    -
encryptgrp          -
%allocated(ese)      0
%allocated(rep)      0
%allocated(std)      0
%allocated(over)     0
```

%virallocated(ese)	-
%virallocated(tse)	-
%virallocated(init)	-
%migrating(in)	0
%migrating(out)	0
numtiers	1
etmanaged	yes
etmigpauseremain	-
etmonpauseremain	-
etmonitorreset	unknown
extsize	21cyl

The extent size of the pool is shown in the last line and some of the measures, such as *available*, is expressed as the number of extents of that size.

If you attempt to add a rank with a different extent size to the pool, you receive an error (as shown in Example 4-5) where a rank with a 1 GiB extent size is to be added to a 16 MiB extent pool.

Example 4-5 Adding a 1 GiB extent rank to a 16 MiB extent pool fails

```
dscli> chrank -extpool p4 r2
CMUN80926E chrank: R2: The rank cannot be assigned to the pool due to mismatched
extent sizes.
```

DS CLI controls

The use of the DS CLI provides more control over the use of extents in an extent pool, as described in Chapter 3.1.1, “Optional ESE repository for thin-provisioned logical volumes” on page 20.

Reserve capacity

It is suggested that some space is reserved when thin-provisioned volumes are used in case you run out of capacity. We enable this reserve and specify the percentage that we want to reserve by using the **chextpool** command (we also can enable this reserve when the extent pool is defined by using the **mkextpool** command), as shown in Example 4-6.

Example 4-6 Reserving capacity in an extent pool

```
dscli> chextpool -extentlimit enable -limit 90 p5
CMUC00001I chextpool: Extent pool P5 successfully modified.
```

```
dscli> showextpool p5
Name                P5-CKD-small
ID                  P5
stgtype             ckd
totlstor (2^30B)    1532
availstor (2^30B)    1532
resvdstor (2^30B)    0
rankgrp             1
numranks            1
numvols             0
status              below
%allocated          10
%available           89
configured          92152
```

allowed	82936
available	82936
allocated	0
reserved	0
configuredCap(MiB/cyl)	1935192
allowedCap(MiB/cyl)	1935192
availableCap(MiB/cyl)	1935192
allocatedCap(MiB/cyl)	0
reservedCap(MiB/cyl)	0
%limit	90
%threshold	15

The allowed and available extents numbers dropped by 10% because we limited the extent pool to 90% of its size; however, the `allowedCap(MiB/cyl)` and `availableCap(MiB/cyl)` still show the *total* that is allowed and available capacity.

Support for overprovisioning controls

Overprovisioning of storage with thin provisioning brings with it the risk of running out of space and losing access to data when applications cannot allocate space that was presented to the servers but is not available on the storage system.

Most IT organizations often establish a policy that dictates the amount of overprovisioning that they allow. It is possible to specify such policy so that a defined overprovisioning level is enforced.

To enforce an overprovisioning limit, specify the **`opratiolimit`** parameter by using the **`mkextpool`** command when creating the extent pool, or with the **`chextpool`** command to modify it later (see Example 4-7).

Example 4-7 Setting the overprovisioning ratio

```
dscli> mkextpool -rankgrp 0 -stgtype fb -opratiolimit 3.5 -encryptgrp 1
test_create_fb
CMUC00000I mkextpool: Extent pool P3 successfully created.
```

```
dscli> chextpool -opratiolimit 3.125 p3
CMUC00001I chextpool: Extent pool P3 successfully modified
```

```
dscli> showextpool p3
```

```
...
%limit 100
%threshold 15
...
opratio 0.76
opratiolimit 3.13
%allocated(ese) 0
%allocated(rep) 0
%allocated(std) 75
%allocated(over) 0
%virallocated(ese) -
%virallocated(tse) -
%virallocated(init) -
...
```

The overprovisioning ratio is based on the formula that is shown in Figure 4-3.

$$\text{Overprovisioning Ratio} = \frac{\text{Allocated Capacity (TP \& standard volumes)}}{\text{Total Capacity} - \text{Overhead capacity} - \text{Reserved Capacity}}$$

Figure 4-3 Overprovisioning ratio

Setting the overprovisioning ratio changes the system behavior to prevent an extent pool from exceeding the overprovisioning ratio in the following situations:

- ▶ Prevent volume creation, expansion, or migration
- ▶ Prevent rank depopulation
- ▶ Prevent pool merge
- ▶ Prevent turning on Easy Tier space reservation

Setting monitoring thresholds

The DS8900F sends out Simple Network Management Protocol (SNMP) warnings when some thresholds are exceeded. The following warning thresholds are available by default:

- ▶ A warning at 15%, meaning that 85% of the extent pool capacity is used. (SNMP must be set up correctly.)
- ▶ A warning at 0%, meaning that the extent pool is 100% filled and no free extents are available.

You also can set a custom-level warning by using the **chextpool** command, as shown in Example 4-8.

Example 4-8 Setting a custom warning threshold

```
dscli> chextpool -threshold 20 p5
CMUC00001I chextpool: Extent pool P5 successfully modified.
```

```
dscli> showextpool p5
Name                P5-CKD-small
ID                  P5
stgtype             ckd
:
:
%limit              90
%threshold          20
:
```

Defining a repository for ESE thin-provisioned volumes

The DS CLI also allows you to reserve capacity for space-efficient volumes by defining a *repository*. A repository is created by using the **mksestg** command. You specify the real capacity to reserve as GiB or blocks for FB extent pools or cyl or mod1 for CKD pools. GiB is the default unit of measure and you can specify the allocation unit by using the **-captype** option. As shown in Example 4-9, we created a 500 GiB repository for the FB extent pool.

Example 4-9 Creating a repository

```
dscli> mksestg -repcap 500 p4
CMUC00342I mksestg: The space-efficient storage for the extent pool P4 has been
created successfully.
```

You can also specify a virtual capacity within an extent pool with large extents. However, this specification does *not* limit the virtual capacity, it pre-allocates only the metadata (large) extents that are required for the specified virtual capacity. For more information, see Chapter 4.1.3, “Creating an extent pool with large extents” on page 39.

If we attempt to define a virtual capacity for an extent pool with small extents, we receive an error (as shown in Example 4-10) because the definition of a virtual capacity for an extent pool with small extents is not allowed.

Example 4-10 Attempting to allocate a virtual capacity for an extent pool with small extents fails

```
dscli> mksestg -vircap 10000 p4
CMUN80932E mksestg: Virtual Space Efficient storage is not supported on 16 MiB or
21 cylinders extent size pools.
```

Setting a threshold for a repository

As thresholds are available for the extent pool, thresholds also are available for the repository. The default thresholds are set at the 15% warning level (when 85% of the repository is used) and at 0% (when 100% is used) and we are at the out-of-space level. In addition to these system-defined thresholds, you can set your own threshold for the repository according to your needs, as shown in Example 4-11.

Example 4-11 Setting a threshold for a repository

```
dscli> chsestg -repcapthreshold 20 p4
CMUC00343I chsestg: The space-efficient storage for the extent pool P4 has been
modified successfully.
```

4.1.3 Creating an extent pool with large extents

In this section, an example of creating an extent pool with large extents is described to show some special features that are available only with large extents.

We create an extent pool and add a CKD rank that is formatted with large extents to the extent pool, as shown in Example 4-12.

Example 4-12 Creating a CKD extent pool with large extents

```
dscli> mkextpool -rankgrp 0 -stgtype ckd P6-CKD-Large
CMUC00000I mkextpool: Extent pool P6 successfully created.
```

```
dscli> mkrank -array a1 -stgtype ckd -extpool p6 -extsize 1113cyl
CMUC00007I mkrank: Rank R3 successfully created.
```

We check the extent pool size by using the **showextpool** command, as shown in Example 4-13 on page 40.

Example 4-13 A CKD extent pool with large extents

```
dscli> showextpool p6
Name                P6-CKD-Large
ID                  P6
stgtype             ckd
totlstor (2^30B)    1793
availstor (2^30B)   1793
resvdstor (2^30B)   0
rankgrp             0
numranks            1
numvols             0
status              below
%allocated           0
%available           100
configured           2034
allowed             2034
available            2034
allocated            0
reserved            0
configuredCap(MiB/cyl) 2263842
allowedCap(MiB/cyl)  2263842
availableCap(MiB/cyl) 2263842
allocatedCap(MiB/cyl) 0
reservedCap(MiB/cyl) 0
%limit              100
%threshold           15
virextstatus         full
%virallocated        0
%viravailable        0
virconfigured        0
virallowed           0
viravailable         0
virallocated         0
virreserved          0
%virextlimit         -
%virextthreshold     -
encryptgrp           -
%allocated(ese)      0
%allocated(rep)      0
%allocated(std)      0
%allocated(over)     0
%virallocated(ese)   -
%virallocated(tse)   -
%virallocated(init)  -
%migrating(in)       0
%migrating(out)      0
numtiers             1
etmanaged            yes
etmigpauseremain     -
etmonpauseremain     -
etmonitorreset       unknown
extsize             1113cyl
```

We now want to create a repository and specify only the virtual capacity to pre-allocate the extents that are needed for the metadata for the specified virtual capacity, as shown in Example 4-14.

Example 4-14 Allocating metadata for a virtual capacity

```
dscli> mksestg -vircap 4000 -captype mod1 p6
CMUC00342I mksestg: The space-efficient storage for the extent pool P6 has been
created successfully.
```

We now check the extent pool space again. As shown in Example 4-15, eight (large) extents are allocated and the available space was reduced.

Example 4-15 Extent pool space after virtual capacity is defined

```
dscli> showextpool p6
Name                P6-CKD-Large
ID                  P6
stgtype             ckd
totlstor (2^30B)    1793
availstor (2^30B)  1785
resvdstor (2^30B)   0
rankgrp             0
numranks            1
numvols             0
status              below
%allocated          0
%available           99
configured          2034
allowed             2034
available           2026
allocated           8
reserved            0
configuredCap(MiB/cyl) 2263842
allowedCap(MiB/cyl)   2263842
availableCap(MiB/cyl) 2254938
allocatedCap(MiB/cyl) 8904
reservedCap(MiB/cyl)  0
%limit              100
%threshold           15
virextstatus        below
%virallocated        0
%viravailable         100
virconfigured        4000
virallowed            4000
viravailable          4000
```

Now, we can better see how much space is available for the volume and pre-allocation of metadata. We can also speed up the volume creation process.

4.2 Creating ESE thin-provisioned volumes

Now that we prepared our extent pools and repositories for thin-provisioned volumes, we create them.

4.2.1 Creating thin-provisioned volumes by using the DS GUI

To define a volume with by using the DS GUI, we browse to the Volume or Volume by Pool panel (see Figure 4-4) and click **Create Volumes**.

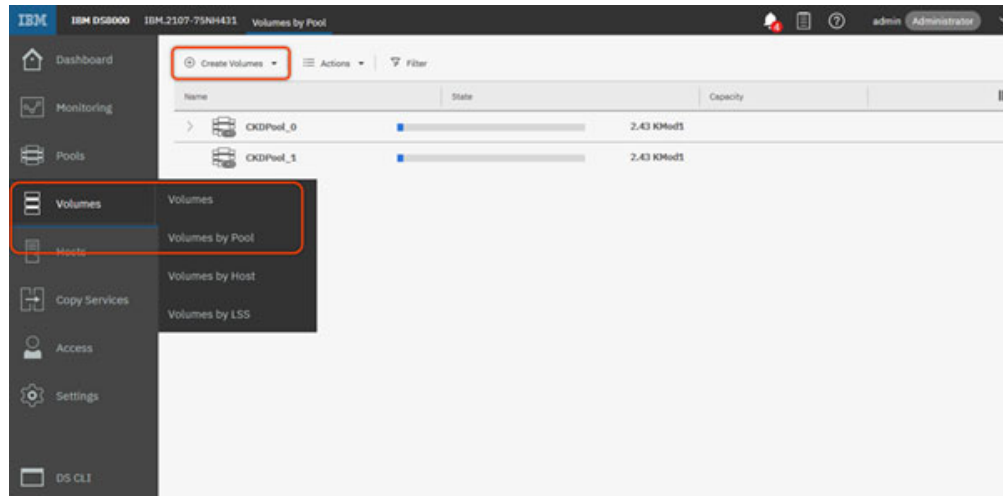


Figure 4-4 Create volumes panel

When the Create Volumes panel opens, the options to create volumes for either **IBM Z** (CKD), **IBM i**, or **Open System** are displayed, as shown in Figure 4-5.

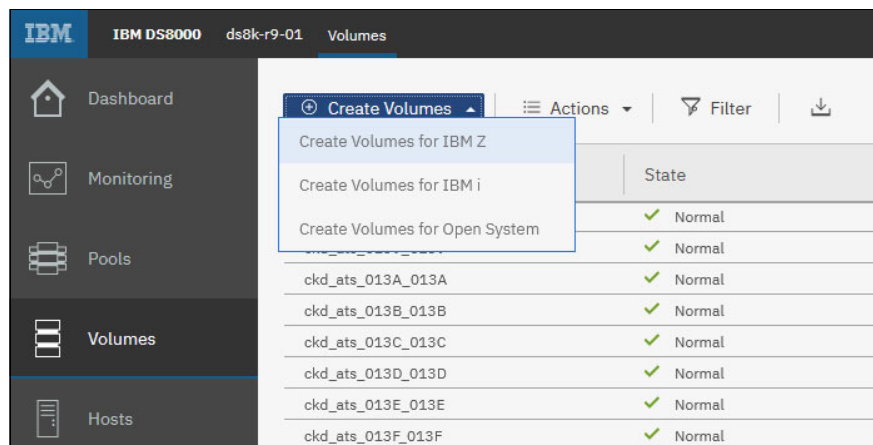


Figure 4-5 New volume creation (R9.3 and higher code levels)

Select the respective operating system.

In Figure 4-6 on page 43, IBM Z is selected and the following fields are filled in:

- Specify the pool pair in which to create volumes.
Default CKD_0 & CKD_1 is used in the example.

- Select the LSS (LCU) range.
LSSs 12+13 is used in the example.
- Specify a name prefix, like *CKDThin*.
- Specify the number of volumes per LSS, and you can specify an address range with these LSSs.
- Define to the nominal capacity of the volumes.
- Select **Provisioning: Thin provisioned (ESE)**, which is your choice when you want to create the volumes as thin.

Figure 4-6 GUI (R9.3+), creating thin volumes (here: for IBM Z)

From there, you can **Save** this first set of volumes and then with *New Volume Set* or the ⊕ symbol to specify further such volume sets. When you have the batch for all volume sets complete, click **Create** to create them together.

4.2.2 Creating thin-provisioned volumes by using the DS CLI

The DS CLI FB volumes are created by using the **mkfbvol** command and CKD volumes are created by using the **mkckdvol** command. The essential parameter that is needed to get thin-provisioned volumes is the Storage Allocation Method parameter **-sam** of both commands and the extent pool where defining the volumes as the extent pool determines the extent size that is used for the volumes.

The default storage allocation method is standard, which allocates fully provisioned volumes. To define ESE thin-provisioned volumes, we must specify the **-sam ese** type, as shown in Example 4-16. We first check that the extent pool that we want to use (pool p4) is composed of small extents. Then, we create thin-provisioned volumes that feature a 1 TiB virtual capacity.

Example 4-16 Creating thin-provisioned ESE FB volumes

```
dsccli> lsextpool -l p4
Name      ID stgtype rankgrp status availstor (2^30B) %allocated available reserved numvols numranks numtiers etmanaged extsize
-----
P4-FB-small P4 fb      0 below      1550      0    99217      0      0      1      1 yes      16MiB

dsccli> mkfbvol -extpool p4 -sam ese -cap 1000 -name TPSEvol_#h 2000-2003
CMUC00025I mkfbvol: FB volume 2000 successfully created.
CMUC00025I mkfbvol: FB volume 2001 successfully created.
```

```
CMUC00025I mkfbvol: FB volume 2002 successfully created.
CMUC00025I mkfbvol: FB volume 2003 successfully created.
```

When we check the extent pool (as shown in Example 4-17), we can see that real capacity was allocated only for the metadata that was required for these virtual volumes. Volume metadata was described in 3.1, “DS8000 logical volumes” on page 18. According to the rules for metadata allocation, four small (16 MiB) extents are allocated for the four volumes.

Example 4-17 Extent pool with some allocated metadata

```
dsccli> showextpool p4
Name                P4-FB-small
ID                  P4
stgtype              fb
totlstor (2^30B)    1550
availstor (2^30B)   1544
resvdstor (2^30B)   0
rankgrp              0
numranks             1
numvols              4
status               below
%allocated           0
%available            99
configured           99217
allowed              99217
available             98813
allocated            404
:
:
```

We created one extent per volume and another 400 small (16 MiB) extents for the virtual capacity of 4000 GiB (one small (16 MiB) extent for each 10 GiB virtual volume size. No space is allocated for user data as of now; only 404 16 MiB extents are for metadata, as shown in Example 4-17.

In a similar way, we now define some CKD thin-provisioned ESE volumes. We assume that the required LCU exists. We check that the CKD extent pool has the extent size that we want to use (small extents with 21 cylinders in our case). Then, we can define thin-provisioned CKD volumes by using the **-sam ese** parameter, as shown in Example 4-18.

Example 4-18 Creating of thin-provisioned ESE CKD volumes

```
dsccli> lsextpool -l p5
Name      ID stgtype rankgrp status availstor (2^30B) %allocated available reserved numvols numranks numtiers etmanaged extsize
-----
P5-CKD-small P5 ckd      1 below      1532      10    82936      0      0      1 1      yes    21cyl

dsccli> mkckdvol -extpool p5 -cap 54 -catype mod1 -sam ese -name TPESEvol_#h 1100-1103
CMUC00021I mkckdvol: CKD Volume 1100 successfully created.
CMUC00021I mkckdvol: CKD Volume 1101 successfully created.
CMUC00021I mkckdvol: CKD Volume 1102 successfully created.
CMUC00021I mkckdvol: CKD Volume 1103 successfully created.
```

We check the extent pool and as shown in Example 4-19, in which 28 small 21 cylinder extents are allocated for metadata and no space yet for user data. We expected this result because one small 21 cyl extent is allocated for each 11130 cylinder or each 10 Mod1. For a Mod54, we need six small extents ($54/10 = 5.4$ rounded up to 6) for each volume. This configuration results in 24 small extents for the four volumes and one small extent for each volume, which results in 28 metadata extents in total.

Example 4-19 Checking the extent pool allocated capacity after CKD ESE volumes are created

```

dscli> showextpool p5
Name                P5-CKD-small
ID                  P5
stgtype             ckd
totlstor (2^30B)    1532
availstor (2^30B)   1532
resvdstor (2^30B)   0
rankgrp             1
numranks             1
numvols              4
status              below
%allocated           10
%available           89
configured           92152
allowed              82936
available            82908
allocated          28
:
```

4.2.3 Re-initializing online space-efficient volumes

You can use the **initfbcol** or **intckdvol** command to re-initialize online space-efficient FB or CKD volumes. These commands include the following options to prevent the accidental re-initialization of volumes that are in use:

Important: All data is lost when these commands are used.

- ▶ An FB volume is considered to be in use if it is participating in a Copy Services relationship or if the volume received any I/O operation in the previous 5 minutes.
- ▶ A CKD volume is considered to be in use if it is participating in a Copy Services relationship or if the IBM Z system path mask indicates that the volume is in a grouped state or online to any host system.

Volume re-initialization is controlled by the **releasespace** and **-force** parameters. Consider the following points:

- ▶ The **releasespace** parameter is used to free up all extents or tracks that are associated with the specified volume so they can be reused by other space-efficient volumes.
- ▶ The **-force** parameter is required in order for the **initfbcol** or **intckdvol** command to re-initialize the specified volume.
- ▶ When both of these parameters are used, the user is prompted for a Y/N response for the command to proceed with the re-initialization process.

An example of the **initfbcol** command is shown in Example 4-20.

Example 4-20 initckdvol command example

```
dscli> initckdvol -action releasespace -force 2300  
CMUC00338W initckdvol: Are you sure that you want to free all extents and lose the  
data associated with CKD volume 2300? [Y/N]:y  
CMUC00340I initckdvol: 2300: The command releasespace has completed successfully.
```

4.3 Summary

As described in this chapter, we saw that we first must provide extent pools with the wanted extent size (small or large extents).

The use of the DS CLI provided more control over the use of extents in an extent pool. We can reserve storage in the extent pool and reserve storage for thin-provisioned volumes and set thresholds.

We then created thin-provisioned volumes with the DS GUI by using the Advanced Custom creation process with which we specified that we wanted to create ESE volumes. We also used the DS CLI and specified the ESE Storage Allocation Method.

The defined thin-provisioned virtual volumes can now be mapped to hosts in the same way as normal volumes.



Managing and monitoring thin-provisioned volumes

This chapter describes the IBM DS8000 monitoring and management capabilities of thin provisioning.

5.1 Management and monitoring capabilities

In the DS8000, thin provisioning and the associated real capacity are manageable at the extent pool level.

5.1.1 Options to reserve space and spare capacity

In 3.1, “DS8000 logical volumes” on page 18, we described an extent pool with its associated real capacity, virtual capacity, and repository capacity. The following options to reserve contingency capacity are available, which apply to real capacity (real extents):

- **Extent limit**

By using this option, the percentage of real extents that can be allocated to logical volumes in an extent pool can be limited. The remaining percentage of real extents is reserved as a contingency buffer for allocation to logical volumes of all types. The extent limit can be adjusted as required by using the DS CLI. The DS GUI does not provide these controls.

- **Reserve ranks**

By using this option, the unallocated capacity of an entire *rank* in an extent pool can be reserved for future use, which makes the reserved capacity unavailable for allocation to logical volumes of all types. This capability works for entire ranks and the remaining unallocated capacity of ranks in an extent pool because all allocations remain unchanged and are not affected by this reserve. The reserve ranks can be dynamically turned on and off by using the **chrank** command. This capability releases some reserve space for use as required by using the following sequence:

- a. Release a rank (or the unused parts of the rank).
- b. Use the free extents; for example, to create a volume.
- c. Reserve the unused space of the rank again.

Important: The larger value of these two options (extent limit and reserve ranks) determines the baseline from which the threshold for real capacity (**-real cap**) is starting to count.

5.1.2 Overprovisioning control

An overprovisioning ratio limit can be set and enforced by the system, as described in “Support for overprovisioning controls” on page 37.

5.1.3 Options to set thresholds and receive warnings

For more information about setting up the thresholds by using DS CLI or DS GUI, see Chapter 4, “DS GUI and DS CLI support for thin provisioning” on page 31.

Thresholds can be set for the real capacity of the entire extent pool and the repository capacity, if a repository is defined.

For the extent pool and repository thresholds, the following system-defined warning thresholds are available:

- A threshold at 15%, which is triggered when 85% of the extent pool or repository are full.
- A threshold at 0%, which is triggered when the extent pool or repository is 100% full.

In addition to these system thresholds, a storage administrator can set a custom threshold at the extent pool or at repository level according to their needs. If a user-defined threshold is set, the extent pool status of whether the used capacity is above or below the threshold is in relation to the user-defined threshold.

Simple Network Management Protocol (SNMP) warnings are triggered when the amount of allocated capacity exceeds any of the specified thresholds. A threshold status changes and can be monitored.

Important: The following status levels and warning messages are listed:

- ▶ Status = 0/Below. Below threshold → Available space → Threshold is good.
- ▶ Status = 1/Exceeded. Warning → Available Space is between Threshold and zero.
- ▶ Status = 10/Full. Space is zero → There is no Available Space in the Extent Pool.

When any of the three thresholds attain a threshold amount, a notification is sent for that particular threshold. No further notifications are sent until the repository capacity changes, for example. If the repository capacity changes and remains above the threshold, another notification might be sent, but no more than one notification every 5 minutes.

You must free capacity in the repository to stop the notifications. If the user-defined threshold is equal to one of the other two fixed thresholds, only one notification is sent (at most once every 5 minutes) for the two equivalent thresholds.

Real capacity threshold

The extent threshold (-extent_threshold_percentage) specifies a percentage of available space, or available extents in the extent pool. It is client-configurable 0% - 100% by using the extent pool command **chextpool1**. The following volume types and virtual capacity can contribute to changes in the number of free extents:

- ▶ Standard volumes when created or deleted.
- ▶ Extent Space Efficient (ESE) volumes when real data extents are assigned as necessary, or when their extents are released again (through initialization, SCSI format unit, or volume deletion).
- ▶ Virtual capacity when created or increased or through deletion. In virtual capacity, associated metadata uses real extents in the extent pool.

Repository capacity threshold

This threshold (- repository_threshold_percentage) specifies the minimum percentage of available physical repository capacity. If the available, free capacity in the repository drops below the percentage value and warnings are sent and the status changes. It is Client-configurable 0% - 100% by using the commands that are related to space-efficient storage, such as **chsestg**.

For both system set thresholds, SNMP warnings are sent every five minutes when exceeded. But the status reported for the repository is based on the threshold that was configured by the Client.

5.1.4 SNMP trap extent pool threshold

SNMP traps can be triggered by the DS8900F. Event trap 223 provides the storage administrator information about extent pool capacity thresholds. A generated event trap 223 is shown in Example 5-1.

Example 5-1 Specific event trap 223

```
Extent Pool Capacity Threshold Reached
UNIT: Mnf Type-Mod SerialNm
      IBM 5333-994 75-XYZ01
Extent Pool ID: P1
Limit: 95%
Threshold: 95%
Status: 0
```

The trap is sent when extent pool capacity thresholds are reached and cause the following changes in the extent status attribute:

- ▶ Extent status is not zero (available space is below threshold) when the first ESE volume is configured.
- ▶ Extent status changes state if ESE volumes are configured in the extent pool.

The extent status attribute is set to a value that is based on the comparison between the extent threshold and the percentage of remaining available real capacity in the extent pool. The extent status value is set as listed in Table 5-1.

Table 5-1 Extent status attribute value

Extent status	Description	Condition
10	%Available Real Cap. = 0	Full: Extent pool full
1	Ext. Threshold >= %Avail. Real Cap. > 0	Exceeded: Threshold is exceeded
0	%Avail. Real Cap. > Ext. Threshold	Below: Below threshold

SNMP traps and their destination (SNMP manager) can be set by using the DS CLI, as shown in Example 5-2.

Example 5-2 Setting and displaying the SNMP trap and destination

```
dscli>chsp -snmp on snmpaddr 9.155.87.211,9.155.66.14,9.145.243.185 -desc "ATS
DS8000 S/N 75-20780" -name ATS_Mainz_20780
Storage-complex IBM.2107-7520781 successfully modified.
dscli> showsp
Name          ATS_Mainz_20780
desc          ATS DS8000 S/N 75-20780
acct          -
SNMP          Enabled
SNMPAddr      9.155.87.211,9.155.66.14,9.145.243.185
emailnotify   Disabled
emailaddr     -
emailrelay    Disabled
emailrelayaddr -
emailrelayhost -
numkssupported 4
...
```

The following software can be used to manage and monitor thin-provisioned volumes:

- ▶ IBM Spectrum™ Control and IBM Storage Insights support thin-provisioned volumes by providing information about the volumes.
An example for Storage Insights that lists the capacity by volume is shown in Figure 5-1.
- ▶ IBM Tivoli® Netcool/OMNIBus can be set up to receive the SNMP traps that are sent by the DS8000 and trigger the necessary actions.
- ▶ Any other system management and monitoring tools that can receive SNMP traps are supported.
- ▶ The **LISTDATA** command can be used in z/OS 2.1 and later. For more information, see Chapter 6.4, “Thin provisioning support in IBM Z” on page 56.

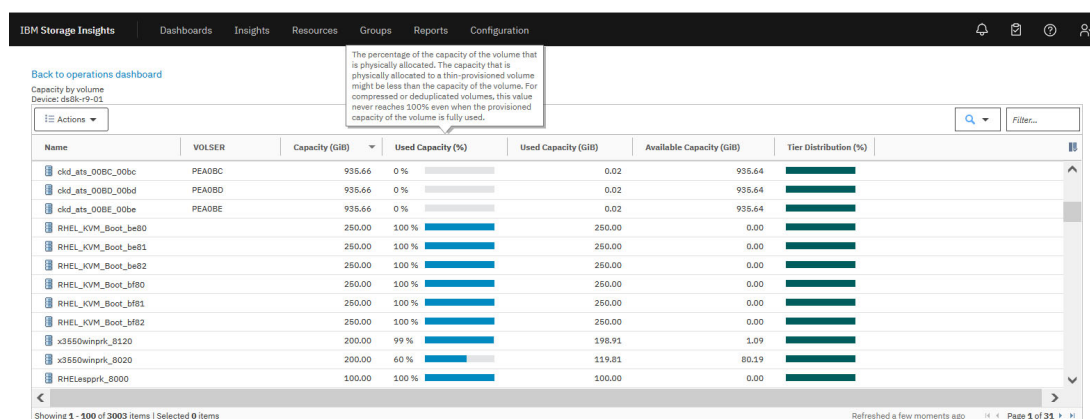


Figure 5-1 IBM Storage Insights listing volumes by capacity

5.1.5 Managing out-of-space situations

Over-provisioning or over-commitment of capacity can result in a situation in which the needed capacity exceeds the available capacity. In the DS8900F implementation of thin provisioning, only the amount of data that is written to disk uses capacity and only for volumes for which real disk capacity exists (write operation by the host is possible).

For write operations that exceed this amount, a write inhibit is sent to the host, which often causes the software to fail. The DS8900F rejects write accesses to the affected volumes until the required extra real capacity is made available for allocation.

The options to provide contingency capacity that are described in 5.1.1, “Options to reserve space and spare capacity” on page 48 and the warnings that are received that relate to the set thresholds can help to avoid an out-of-space situation. However, when these options are exhausted, the following options are available to release space on ESE volumes:

- ▶ Issue SCSI format unit command from the host.
- ▶ Delete or remove some volumes.



Use of thin-provisioned volumes in IBM Z

This chapter describes some features that are available in IBM Z to manage thin-provisioned volumes and extent pools.

This chapter includes the following topics:

- ▶ 6.1, “Thin-provisioned Count Key Data devices” on page 54
- ▶ 6.2, “Advantages of using thin-provisioned volumes in z/OS” on page 55
- ▶ 6.3, “Migrating to thin-provisioned volumes” on page 55
- ▶ 6.4, “Thin provisioning support in IBM Z” on page 56
- ▶ 6.5, “DFSMSdss SPACEREL command” on page 58
- ▶ 6.6, “Thin-provisioning and copy services” on page 60

6.1 Thin-provisioned Count Key Data devices

Count Key Data (CKD) devices can be defined as thin-provisioned Extent Space Efficient (ESE) devices, as shown in Figure 6-1 Thin-provisioned CKD volume. Thin-provisioning also allows you to over-provision capacity.

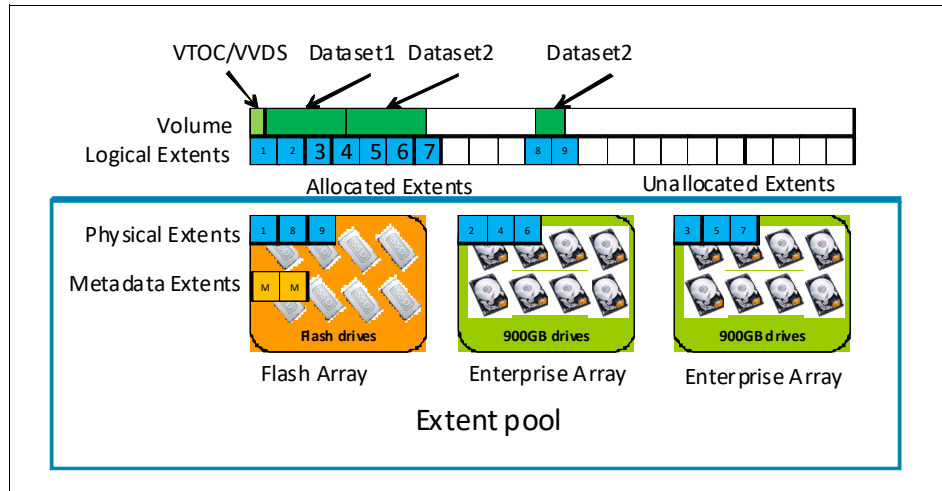


Figure 6-1 Thin-provisioned CKD volume

You can define CKD virtual volumes in which the total virtual capacity (the sum of the virtual capacity of all volumes) is larger than the physical capacity that is available in the extent pool. However, this definition requires that you monitor the free capacity in the extent pool. The means available in IBM Z to set this definition are described in this chapter.

Notes: Consider the following points:

- ▶ Over-provisioning capacity to a large extent results in some amount of metadata. For best performance, some flash capacity is needed in each pool. Metadata is placed in the highest tier.
- ▶ Thin-provisioned ESE volumes can be used as FlashCopy target volumes.
- ▶ Linux on IBM Z does not support thin-provisioned CKD devices because the formatting of each track results in the device becoming fully provisioned.

The **DFSMSdss SPACEREL** command releases physical space on SMS and non-SMS volumes. It also releases space on CKD Primary and Secondary volumes with different extent sizes. Note that the space released on one volume might not match that of the other volume.

Space also is released on FlashCopy target volumes when the target volume is thin-provisioned. A space release is performed on the target of a Metro Mirror or Global Copy when the relationship is established if the source and target are thin-provisioned.

If DFSMSHsm is used for your space management to migrate data according to policy to higher migration levels to provide free space on your primary volumes if thresholds are exceeded, it is advisable to adjust these thresholds and lower them.

Compared to volumes managed by DFSMSHsm, thin-provisioning makes more sense for volumes with a more static data allocation; for example, database volumes. However, if you set the threshold correctly, you can use DFSMSHsm space management and thin-provisioning together; for example, when you want all volumes to be thin-provisioned.

Tip: Adjust your DFSMSHsm migration thresholds when large thin-provisioned volumes are used.

6.2 Advantages of using thin-provisioned volumes in z/OS

Thin-provisioning can make storage administration easier. You can provision large volumes when you configure a new DS8900F. You do not have to manage different volume sizes, when to use a 3390 Model 1 size, a Model 9 or better a Model 27, and so on. You can make all volumes large and of the same size.

At times, a volume or device address is required to communicate with the control unit, such as the utility device for z/OS Global Mirror (zGM or XRC). Such a volume can include a minimum capacity. With thin-provisioning, you still can use a large volume because less data is written to such a volume, its size remains small, and no storage capacity is wasted.

For many z/OS customers, migrating to larger volumes is a task they avoid because it involves too much work. As a result, many customers have too many small 3390 Model 3 volumes. With thin-provisioning, you can define large volumes and migrate your data from your storage system to a DS8900F that is defined with thin-provisioned volumes and likely use even less space. Most migration methods facilitate copying small volumes to a larger volume. You refresh the VTOC of the volume to recognize the new size.

6.3 Migrating to thin-provisioned volumes

In the current implementation, a volume in place cannot be converted from fully provisioned to thin-provisioned.

FlashCopy can be used to convert a volume. Target space is released when FlashCopy is used with a full background copy from a fully provisioned volume to a thin-provisioned volume. Then, only the data is copied and not the empty space.

FlashCopy is practical to use for a few volumes, but it is not the best choice for migrating data from an old DS8000; for example, with fully-provisioned volumes, migrating to a DS8900F with thin-provisioned volumes.

You can use Metro Mirror or Global Copy to copy your volumes from an old DS8000 to a DS8900F.

However, with R8.1.x and before, the available Metro Mirror and Global Copy options are fully provisioned-to-fully provisioned and thin-provisioned-to-thin-provisioned only. In this case, you might consider first using Metro Mirror or Global Mirror to move the data from the old DS8000 to fully provisioned temporary volumes on the DS8900F and then use the FlashCopy method to copy all volumes to thin-provisioned volumes in one step or one after the other.

With DS8900F, you can use any-to-any, meaning Metro Mirror and Global Mirror to move data from fully provisioned to thin-provisioned volumes.

A better method is to use IBM Transparent Data Migration Facility (TDMF for z/OS) as the migration method. TDMF provides online data migration capabilities at the volume level. It includes the FASTCOPY option that copies only allocated cylinders when volumes are migrated. TDMF for z/OS dynamically detects ESE target volumes and enforces the FASTCOPY option.

Important: The FASTCOPY option is not available for z/VM volumes. The default is NOFASTCOPY for z/VM.

Another option is to use IBM z/OS Data Set Mobility Facility (zDMF).

Logical data set level migration also is possible for thick-to-thin migration.

6.4 Thin provisioning support in IBM Z

z/OS and its components were enhanced to support thin-provisioning.

- ▶ Base Support
 - z/OS V2R3 through V2R5 base
- ▶ Volume-Level Space Release
 - z/OS V2R3 through V2R5 base
- ▶ Space Release MINCYLs Enhancement (OA55666)
 - PTFs on z/OS V2R2 and V2R3
 - z/OS V2R4 and V2R5 base
- ▶ z/VM APAR VM66098
 - PTFs on V7R1 and V7R2

Important: Check FIXCAT or PSP buckets for the latest updates.

The following enhancements are provided:

- ▶ DFSMS provides pool utilization alerts for extent pools.
- ▶ DFSMS with z/OS 2.2 provides storage group utilization alerting, which can be helpful with thin-provisioning. A partial Storage Group display is shown in Example 6-1 on page 56.

Example 6-1 D SMS,SG(ALL),LISTVOL command partial output

```
IGD002I 02:25:40 DISPLAY SMS 582
...
STORGRP TYPE SYSTEM= 1
DB2SY127 POOL +
SPACE INFORMATION:
TOTAL SPACE = 52326MB USAGE% = 13 ALERT% = 0
TRACK-MANAGED SPACE = 52326MB USAGE% = 13 ALERT% = 0
```

- ▶ IDCAMS reports are enhanced to show thin-provisioning statistics.
- ▶ DFSMSdss Move request that DS8000 pre-allocates extents when FlashCopy is used for migration to prevent data loss if the storage pool runs out of extents.

When extent pool thresholds are exceeded, storage pool utilization alert messages are issued, as shown in Example 6-2 on page 56.

Example 6-2 Extent pool utilization alert messages

```
IEA499E dev,volser,epid,ssid,pcnt EXTENT POOL CAPACITY THRESHOLD: AT pcnt%
CAPACITY REMAINING
```

IEA499E dev,volser,epid,ssid,pcnt EXTENT POOL CAPACITY WARNING: AT pcnt% CAPACITY REMAINING

IEA499E dev,volser,epid,ssid,pcnt EXTENT POOL CAPACITY EXHAUSTED

IDCAMS LISTDATA commands are available to check the used capacity of extent pools and volumes. Example 6-3 on page 57 shows the use of the **VOLSPACE** option of the **LISTDATA** command to show how much capacity is used by volumes and the type (Standard or ESE) of volume that is used.

Example 6-3 New LISTDATA VOLSPACE command

LISTDATA VOLSPACE VOLUME(IN9029) UNIT(3390) ALL LEGEND

2107 STORAGE CONTROL						
VOLUME SPACE REPORT						
STORAGE FACILITY IMAGE ID 002107.981.IBM.75.0000000DKA61						
SUBSYSTEM ID X'2400'						
.....STATUS.....						
DEVICE	VOLSER	CAPUSED (CYL)	CAP (CYL)	EXTENT POOL ID	SAM	
900F	IN900F	3339	3339	0000	STD	Large
902A	IN902A	2226	3339	0000	ESE	Extents

2107 STORAGE CONTROL						
VOLUME SPACE REPORT						
STORAGE FACILITY IMAGE ID 002107.981.IBM.75.0000000DKA61						
SUBSYSTEM ID X'2403'						
.....STATUS.....						
DEVICE	VOLSER	CAPUSED (CYL)	CAP (CYL)	EXTENT POOL ID	SAM	
9127	INF45	21	1113	0001	ESE	Small
9129	INF49	21	3339	0001	ESE	Extents
TOTAL NUMBER OF EXTENT SPACE EFFICIENT VOLUME(S):						3
TOTAL NUMBER OF STANDARD VOLUME(S):						1

Example 6-4 on page 57 shows the use of the **LISTDATA EXTENTPOOLCONFIG** command that provides information about the size of an extent pool and how much is in use (allocated).

Example 6-4 LISTDATA EXTENTPOOLCONFIG command

listdata extentpoolconfig extentpoolid(0) volume(px8610) unit(3390)

```
2107 STORAGE CONTROL
EXTENT POOL CONFIGURATION REPORT
STORAGE FACILITY IMAGE ID 002107.994.IBM.75.0000000LN001

.....EXTENT POOL ID 0000 SUMMARY.....
REPOSITORY FULL WARNING PERCENTAGE:          0
EXT POOL FULL WARNING PERCENTAGE:             15
EXTENT POOL STATUS
FIXED BLOCK EXT POOL:                          NO
REPOSITORY CONFIGURED:                         NO
EXTENT POOL AT WARNING PERCENTAGE:             NO
EXTENT POOL FULL:                             NO
...EXTENT POOL 0000 DETAILED REPORT....
EXTENT POOL REPOSITORY STATUS
REPOSITORY AT WARNING PERCENTAGE:              NO
```

REPOSITORY FULL:		NO
	SIZE	ALLOCATED
EXTENT POOL	60224241	47141367
REPOSITORY	0	0

6.5 DFSMSdss SPACEREL command

z/OS V2R2 and above supports the DFSMSdss command **SPACEREL**. This command can be used against SMS or non-SMS volumes or Storage Groups to release physical space that was allocated but not released when the data was logically deleted.

The use of the **SPACEREL** command identifies the logical free space tracks on the specified ESE volume and attempts to release the related physical space back to the extent pool.

For more information about APAR OA50675, see:

- ▶ The IBM Support website at: <https://www.ibm.com/support/pages/apar/OA50675>
- ▶ The IBM Documentation website at:
<https://www.ibm.com/docs/en/zos/2.5.0?topic=v2r3-using-dfsmsdss-enhancements>

The DFSMSdss SPACEREL command also supports Global Mirror, Global Copy, Cascaded PPRC, and Multi-target PPRC Primary and Secondary volumes.

Note: The Primary and Secondary volumes might have different extent sizes. Therefore, different amounts of space might be released on the primary and that of the secondary.

Global Copy and Global Mirror

When the SPACEREL command is used, Global Copy and Global Mirror preserve write-order consistency between the primary and secondary volumes.

Figure 6-2 Global Copy / Global Mirror Space Release Process shows this process.z

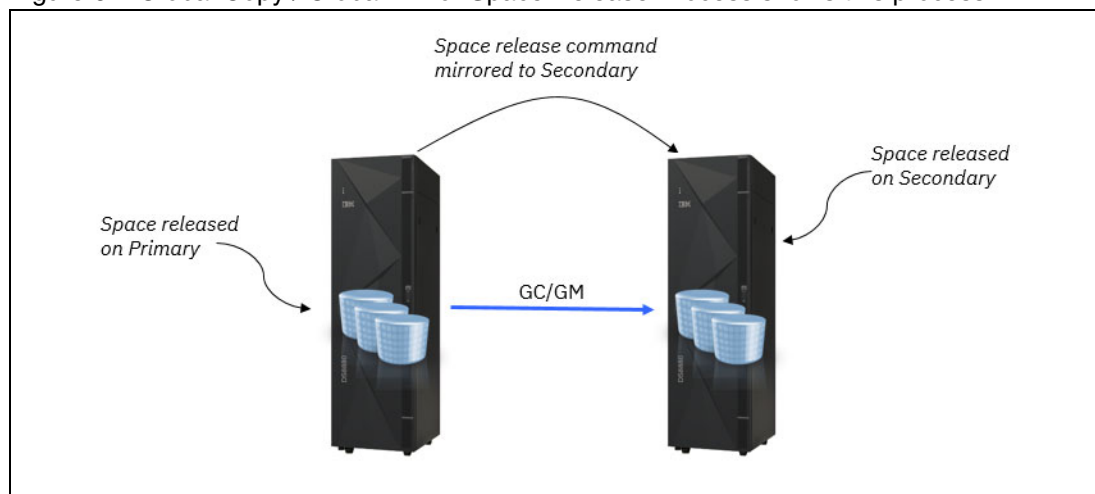


Figure 6-2 Global Copy / Global Mirror Space Release Process

In a Global Copy environment, space is released asynchronously to the command. The SPACEREL command is first sent to the secondary for processing, then space is released on the primary.

Global Mirror coordinates the releasing of space with the formation of Consistency Groups (CGs).

- ▶ Space is released on the primary at the completion of the current CG.
- ▶ Space is released on the secondary at the completion of the next CG (FlashCopy commit).

In order to minimize any impact to the Recovery Point Objective (RPO), a time limit of one second exists for the release-space task on the secondary. Whatever space that is not released during 1 CG will be carried forward to following CGs. It might take multiple CGs before all space is released.

Cascaded PPRC and Multi-Target PPRC operate the same way as Global Copy or Global Mirror. For examples, see Figure 6-3 Cascaded PPRC, and Figure 6-4 Multi-Target PPRC Space-Release Process.

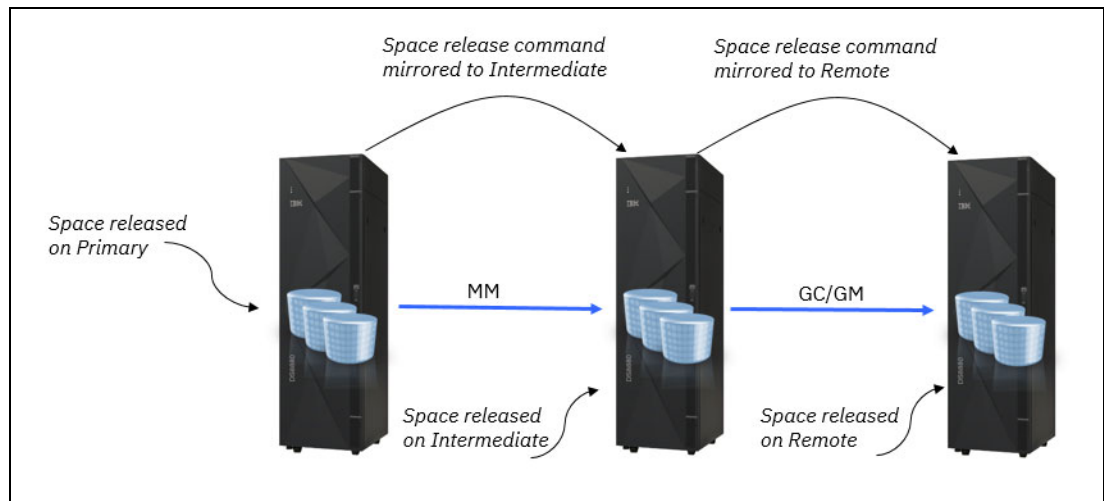


Figure 6-3 Cascaded PPRC

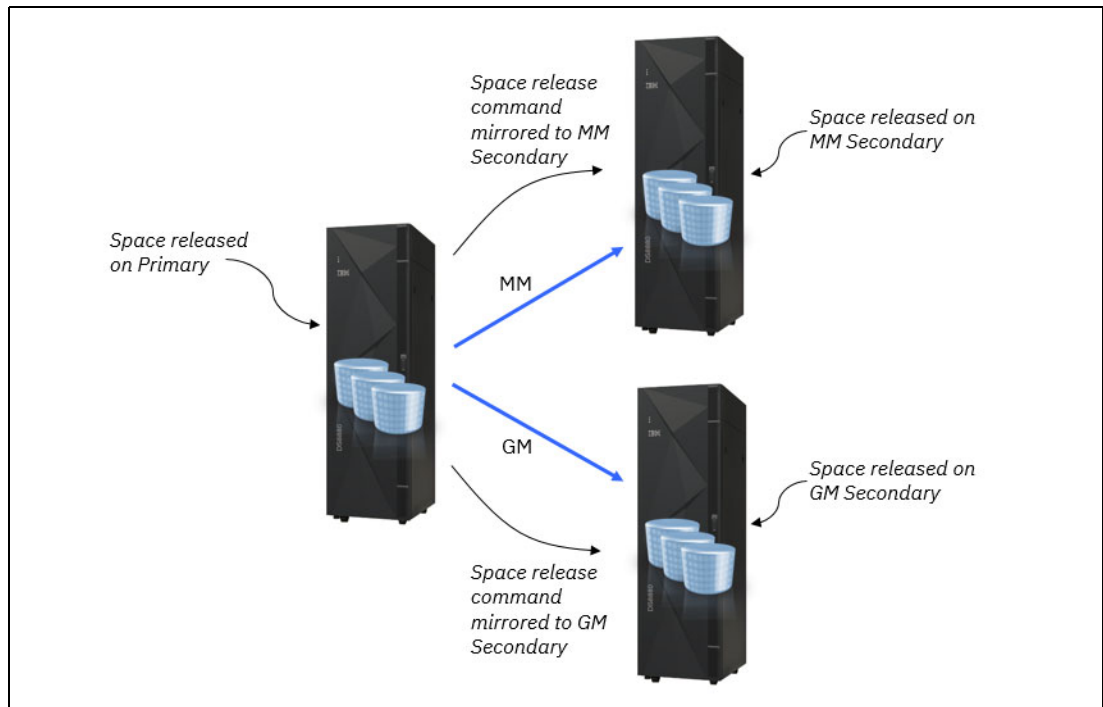


Figure 6-4 Multi-Target PPRC Space-Release Process

6.6 Thin-provisioning and copy services

This section describes some aspects of thin-provisioning and copy services.

FlashCopy

FlashCopy is supported between any type of volume, standard fully provisioned, and ESE thin-provisioned volumes. When the `mkflash` command of the DS CLI is used, the `-tgtse` option must be specified.

If the target volume is an ESE volume, space is released during the FlashCopy establishment process. If a FlashCopy is withdrawn, space also is released at the target volume.

ESE volumes as FlashCopy target volumes are suited for FlashCopy NOCOPY operations in which the FlashCopy dumps the data to tape from the FlashCopy target and then, withdraws the FlashCopy relation.

Metro Mirror and Global Copy

Metro Mirror and Global Copy support full provisioned to full provisioned, thin-provisioned to thin-provisioned, as well as a mix. However, a fully-provisioned volume that is replicated to a thin-provisioned volume results in a fully-provisioned volume. If migration from full to thin-provisioned is desired, see Chapter 6.3, “Migrating to thin-provisioned volumes” on page 55 for more information.

If the source is already a thin volume, this reduces copy bandwidth because fewer tracks need to be copied initially.

If the target is an ESE thin-provisioned volume, space is released at the target and only the used extents are copied from source to target.

You can use the Remote Pair FlashCopy (also known as Preserve Mirror) for thin-to-thin-provisioned volume mirrors; however, space is not released at the remote FlashCopy target.

Restriction: Preserve mirror does not release space on the secondary Metro Mirror FlashCopy target.

z/OS Global Mirror

z/OS Global Mirror (zGM; formerly XRC) primary volumes can be thin-provisioned ESE volumes. zGM secondary volumes also can be thin-provisioned ESE volumes; however, they become fully provisioned.

Attention: zGM secondary volumes become fully provisioned.

Global Mirror

The Global Mirror FlashCopy Journal volumes at the secondary site are a key use case for ESE thin-provisioned volumes (see Figure 6-5 Global Mirror with ESE volumes). ESE volumes reduce the extra space that is required for this solution.

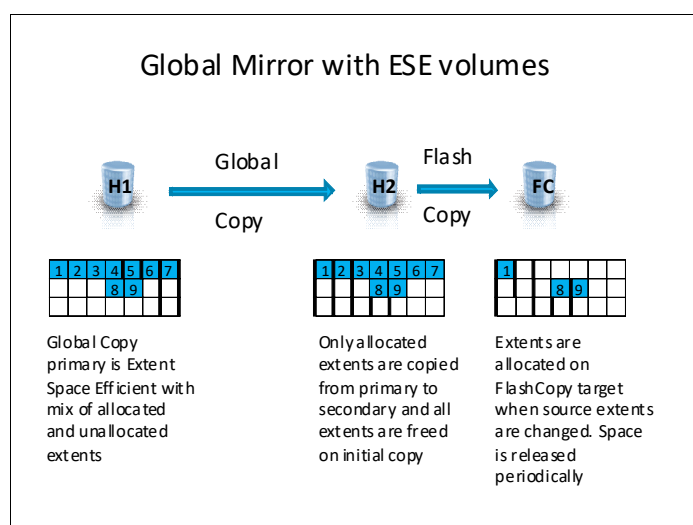


Figure 6-5 Global Mirror with ESE volumes

However, too much overhead can occur for Global Mirror to free extents whenever a consistency group is formed. Therefore, Global Mirror periodically performs a space release while it is running and considers the activity of the storage systems. For more information, see “Global Copy and Global Mirror” on page 58.

The space that is required for the FlashCopy target volumes is higher than with certain previous implementations (TSE). Plan for approximately 20–30% FlashCopy capacity as a precaution when Global Mirror in normal operation is used.

Related publications

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

IBM Redbooks

For more information, see the following IBM Redbooks publication:

- ▶ *IBM DS8900F Architecture and Implementation: Updated for Release 9.3*, SG24-8456

Other publications

The following publications are also relevant as further information sources:

- ▶ *IBM DS8900F Introduction and Planning Guide*, SC27-9560
- ▶ *IBM DS8900F Product Guide Release 9.3*, REDP-5554
- ▶ *IBM DS8000 Series Command-Line Interface User's Guide*, SC27-9562

Online resources

The following websites also are relevant as further information sources:

- ▶ Documentation for the IBM DS8900F at IBM Documentation:
<https://www.ibm.com/docs/en/ds8900>
- ▶ System Storage Interoperation Center (SSIC):
<https://www.ibm.com/systems/support/storage/ssic>

How to get Redbooks publications

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications, and additional materials, and order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

- ▶ IBM Support and downloads
ibm.com/support
- ▶ IBM Global Services
ibm.com/services



REDP-5343-02

ISBN 0738460702

Printed in U.S.A.

Get connected

