White Paper

VPLEX-VMAX Multiple Masking Views Support in EMC ViPR Controller 2.2 and higher

Author
Tom Watson

Abstract

This white paper explains the EMC ViPR Controller 2.2 functionality that allows the generation or discovery and use of multiple masking views on an EMC VMAX or VNX array for hosting VPLEX volumes.

November 2015

Part Number H14006

Table of Contents

Executive summary

This document describes EMC® ViPR® Controller 2.2 and higher functionality that allows the generation or discovery of multiple masking views (MVs) on a VMAX® or storage groups on a VNX® for hosting the VPLEX® volumes. This work is most beneficial on the VMAX, which will be described in detail here. There is some benefit to applying a similar technique to the VNX, which will be mentioned, but is not the focus of this document. Supporting other arrays as a backend to the VPLEX is not discussed here.

The motivations for providing this support include:

- Some customers desire explicit control of how many MVs are to be used between the VPLEX and a VMAX.
- Other customers want automatic generation of an appropriate number of MVs used between the VPLEX and a VMAX. The ViPR 2.2 and higher code does this based on the number of VPLEX back-end ports and VMAX ports that are available for use.

The details of how to set up such a configuration manually are described in this document. There is a substantial amount of manual configuration required to enable support for multiple MVs. You should read this document completely and adhere to all its recommendations and requirements to ensure a successful configuration.

Alternately, the ViPR Controller 2.2 code automatically generates one or more MVs as appropriate, based on your configuration. The details of what to expect for the automatically generated configurations are also described.

Audience

This white paper is written for data center administrators and ViPR system administrators.

VMAX Limitations

There are two principal scalability reasons that more than one MV is required. These are:

- The VMAX has a limit that only 4096 volumes can exist in a MV. If cascaded Initiator Groups are used with separate MVs, this limit can be avoided.
- There is a limit that only 4096 volumes can be processed by a VMAX CPU. Each pair of ports like FA7E:0 and FA7E:1 share a single VMAX CPU. Further this CPU limitation is affected by whether or not an exported volume is a meta volume; if so each meta component of the exported volume counts towards the VMAX CPU

volume limit. Also, if the same volume is referenced from both ports in a CPU, it only counts once towards the limit.

Overview of ViPR Controller Provisioning Operations

When ViPR Controller receives a request to provision a VPLEX local virtual volume, the following operations happen:

1. ViPR Controller creates a volume on a storage array to hold the data for the virtual volume. This volume could be termed the *backing volume*, and the array the *backing array*, as they provide the storage for the virtual volume.
2. ViPR Controller reads the existing MVs off the backing array that contain the initiators which are the VPLEX back-end ports. (The VPLEX back-end ports are used exclusively to handle traffic to backing arrays. The backing arrays see the VPLEX back-end ports as initiators.)
    - If there one or more existing MVs that can be validated to meet ViPR Controller requirements, the volume(s) will be added to the validated MV with the lowest volume count.
    - Otherwise, ViPR Controller will automatically create multiple cascaded Storage Groups and MVs to hold backing volumes for a particular VPLEX on the backing array as they are needed.
    - If ViPR Controller created the MV that was used, ViPR Controller will ensure that zones are created for the Initiator to Target mappings in the MV if the ViPR Controller `auto_san_zoning` boolean in the Virtual Array is true. If ViPR Controller used an existing MV off the backing array that it did not create, no zoning will be attempted (because the zoning should have also been manually configured by the administrator.)
3. ViPR Controller discovers and claims the backing volume in the VPLEX cluster and uses it to build a VPLEX virtual volume.

In more complicated VPLEX provisioning cases (as determined by the virtual pool parameters), ViPR Controller may create more than one backing volume per virtual volume. For example, a backing volume on two different arrays within the two VPLEX clusters is used to construct a distributed virtual volume.

ViPR Controller Restrictions on Masking Views for VPLEX to VMAX Communication

When ViPR Controller receives a request to create virtual volumes using storage provided by a VMAX, it reads the existing MVs on the VMAX and determines if there are any suitable MVs in which it can place the volume. This determination is made each time a backing volume is created on the VMAX and needs to be exported to the VPLEX.

Although ViPR Controller reads and checks the MV, no attempt is made to read the zoning information that would map the VPLEX back-end ports to the VMAX ports unless ViPR Controller created the MV.

ViPR Controller imposes certain restrictions on what it considers a valid MV, based on the VPLEX and VMAX best practices. Here are the validations it performs for each MV, whether it was externally created or internally generated:

Table 1. Masking View Validations Performed by ViPR Controller

No.	Validated Restrictions	Reason for Restriction
1	A MV *must* contain at least two initiators from each VPLEX director.	VPLEX best practice dictates this for redundancy.
2	A MV *must* have at least two usable array ports. A warning is issued if there are less than four usable ports. For a port to be usable, it must be: • On a Network that connects the VPLEX initiators and storage array, and • Assigned to the Virtual Array in which the volume(s) are being created.	If there are fewer than two ports, there is no redundancy. If there are fewer than four ports, then the redundancy is sub-optimal because the MV cannot provide the optimum of four paths per director.
3	A MV *must not* have initiators from both VPLEX clusters. Only initiators from directors on one of the VPLEX clusters are allowed.	If both clusters have initiators in the MV, then volumes created on the VMAX will be visible to both VPLEX clusters, which will cause ViPR Controller provisioning errors.
4	The MV name should not contain the characters `NO_VIPR` (case insensitive). MVs with `NO_VIPR` in their name are interpreted to mean that the administrator desires them to be ignored by ViPR Controller.	This allows the administrator to set up special MVs for cross-connected VPLEX metadata or logging volumes. It also allows the administrator to direct ViPR Controller to no longer assign volumes to an existing MV.

There are other restrictions on MVs that ViPR Controller does not enforce. These restrictions *must* be obeyed by the administrator:

Table 2. Required Masking View Restrictions Not Validated by ViPR Controller

No.	Unvalidated Restrictions	Reason for Not Validating
1	There *must not* be more than four paths from any director to the backing array. This is VPLEX best practice. All ViPR generated MVs will adhere to this requirement.	ViPR Controller does not read zoning for manually created MVs. This allows the administrator the most freedom in how the zones are created, but places more responsibility for verifying configuration correctness on the administrator.
2	The MV *must* provide redundant connectivity between every VPLEX director	Since ViPR Controller does not read the zoning information for manually

No.	Unvalidated Restrictions	Reason for Not Validating
	and the back-end array. If volumes are added to MVs that are manually created but do not provide the required connectivity, provisioning of the virtual volume will fail.	created MVs, this cannot be validated. The administrator is responsible for ensuring connectivity before attempting provisioning.
3	A cascading set of Storage Groups *must* be created that contains each manually created MV. The parent Storage Group can be named `xxx` (where `xxx` is any acceptable name), but the child Storage Group *must* be called `xxx_SG_NonFast`. This allows the ViPR Controller FAST processing logic to put volumes without a FAST policy in the `NonFast` Storage Group. ViPR Controller will add additional child Storage Groups for each FAST policy that is applied to volumes. If you do not use this naming convention, ViPR Controller may not be able to properly provision FAST virtual volumes using the backing array.	Not validated at this time.
4	An Initiator *should not* be in multiple Initiator groups.	Not validated at this time.
5	The same array port *should not* be used in multiple MVs.	Not validated at this time.

Example Validation Messages

When ViPR Controller validates Export Masks, it logs details about each mask and its validity in the `controllersvc.log`. Here are some sample messages with explanations:

```
controllersvc.log.20140208-122359:2014-02-08 10:57:24,346 [pool-5-
Searching for existing ExportMasks between VPLEX  VPLEX _device (VPLEX
+FNM00114300288:FNM00114600001) and Array SYMMETRIX+000195701573
(SYMMETRIX+000195701573) in Varray urn:storageos:VirtualArray:93b108fd-
816e-4660-8810-f0ebf64f7a4c:
```

This indicates it is searching for existing masks, which are listed below.

```
Mask VPLEX 154_no_vipr (urn:storageos:ExportMask:127ac0e3-cf89-465c-
a536-903318f8b821:) Externally created

Mask VPLEX 154BadMixedClusters (urn:storageos:ExportMask:0b093342-854d-
452d-9e39-620096356f83:) Externally created

Mask VPLEX 154A (urn:storageos:ExportMask:c549d5f9-e172-4a32-b06f-
eb6afb15edbc:) Externally created

Mask VPLEX 154C (urn:storageos:ExportMask:8c86bb7e-6326-4a7b-8bf0-
14a876461050:) Externally created

Mask Vpex154B (urn:storageos:ExportMask:e8782b18-1894-4c8c-bda6-
c8a2da60cdaf:) Externally created

Validating ExportMask VPLEX 154_no_vipr
```

This indicates it is validating a specific mask.

```
Warning: ExportMask VPLEX 154_no_vipr has only 2 target ports (best
practice is at least four)

ExportMask VPLEX 154_no_vipr disqualified because the name contains
NO_VIPR (in upper or lower case) to exclude it
```

This indicates the validation failed and why.

```
Validating ExportMask VPLEX 154BadMixedClusters

Warning: ExportMask VPLEX 154BadMixedClusters has only 2 target ports
(best practice is at least four)

ExportMask VPLEX 154BadMixedClusters disqualified because it contains
wwns from both VPLEX  clusters

Validating ExportMask VPLEX 154A

Warning: ExportMask VPLEX 154A has only 2 target ports (best practice is
at least four)

Validation of ExportMask VPLEX 154A passed; it has 3 volumes
```

This indicates that the validation of an Export Mask succeeded.

```
Validating ExportMask VPLEX 154C

Warning: ExportMask VPLEX 154C has only 2 target ports (best practice is
at least four)

Validation of ExportMask VPLEX 154C passed; it has 0 volumes

Validating ExportMask Vpex154B

Warning: ExportMask Vpex154B has only 2 target ports (best practice is
at least four)

Validation of ExportMask Vpex154B passed; it has 1 volumes

Returning new ExportGroup VPLEX
_FNM00114300288:FNM00114600001_000195701573_f92d981d

Returning ExportMask VPLEX 154C (urn:storageos:ExportMask:8c86bb7e-6326-
4a7b-8bf0-14a876461050:)
```

This indicates which ExportMask was selected for use.

Automatic Generation of Multiple Masking Views on the VMAX

You should decide as part of initial system installation if you want ViPR Controller to generate the MVs for use by the VPLEX on your VMAX, or if you want to do it manually. If you decide you want to do it manually, skip to the section Setting Up the Pre-Defined Masking Views for instructions.

There are advantages to either approach. Here are the benefits of Automation Generation:

- The administrator does not have to be concerned about too many of the configuration details that would be required to manually set up a configuration.

- ViPR Controller will set up a reasonable default configuration once as part of the first VPLEX volume that created with the VMAX as a target. A different configuration will be set up for each VMAX or VNX. The number of MVs that are created is dependent on the number of ports available to connect the VPLEX and VMAX (or VNX).
- If the automatically created MVs need to be adjusted (for example to add additional Storage Ports), or additional MVs need to be created, this can subsequently be done manually by the System Administrator using Unisphere.

Advantages of Manually Setting Up the MVs:

- The administrator has explicit control of the number of MVs that are set up, as well as the amount of connectivity and redundancy provided by each MV. This may be best for an expert administrator.
- The administrator has explicit control over the Port Grouping that is used for each MV.
- If the MVs need to be adjusted (for example to add additional Storage Ports), or additional MVs need to be created, this can be done manually be the System Administrator using Unisphere.

ViPR Controller's Decision to Create or Use Externally Created Masking Views

When a request is made to create a VPLEX virtual volume, and ViPR Controller has selected the storage array(s) that will provide the backing storage for the virtual volume, ViPR Controller reads the existing MVs (or VNX Storage Groups) off the array as described above, and attempts to validate them. ViPR Controller determines if there are any valid MVs that were externally created, as well as if there are any valid MVs that ViPR Controller created.

If there are no valid MVs on the array, ViPR Controller will run its algorithm to plan the automatically generated MVs, described below. This algorithm generates a plan for one or more MVs and persists it in the ViPR Controller database. This algorithm is only run once; it generates the plan for all the MVs at one time so as to balance resource usage among them.

The MVs are saved as Export Groups in the VPLEX System Project as show below. They will include ViPR Controller planned MVs, ViPR Controller generated MVs, as well as those that were externally created. Here is a sample representation:

```
python # export_group list VPLEX+FNM00114300288:FNM00114600001

name                                     active   #vols    #ini     id

VPLEX_0288_0001_CL1-c12133ae_vmax1573    YES      1        8
urn:storageos:ExportGroup:b0c44d11-d160-427a-a914-88ed4742d323:vdc1

VPLEX_0288_0001_CL1-eb6a2fcb_vmax1573    YES      1        8
urn:storageos:ExportGroup:b1d0592b-0c1c-4512-8747-a0b31c1a9db7:vdc1

VPLEX_0288_0001_vmax1573-c7d0144d        YES      1        7
urn:storageos:ExportGroup:ee353531-5886-4509-b637-a3a4cb813c14:vdc1
```

This shows three ViPR Controller Export Groups. Each contains an Export Mask which represents a MV or a VNX Storage Group. Two of these were automatically created (those with CL1) and the other was externally created.

Once there are available MVs for use, the volume(s) that need to be exported on this array for the virtual volume create request are added to the MV with the least number of volumes. If a ViPR Controller planned MV is selected, it will be created on the array as part of the provisioning process. Otherwise, volumes will just be added to the existing MV (ViPR Controller generated or externally created).

How ViPR Controller Plans Masking Views for the VMAX

ViPR Controller determines the Virtual Array that is being used for the virtual volume create request. It also determines what VPLEX cluster (in a metro configuration) is being used to handle the request.

ViPR Controller then determines the available VPLEX back-end ports and array (front-end) ports that can be used in the Virtual Array. The VPLEX back-end ports will be referred to as Initiators, as that is the role they play from the storage array's point of view.

In order to satisfy VPLEX best practice, at least two Initiators must be usable from each VPLEX director in the cluster. If that requirement cannot be met, no MVs can be created.

The array ports are divided up by the Network they are on. To be usable, they must be on a Network that has Initiators from the VPLEX. Any ports not on a common network with VPLEX Initiators are discarded.

Additionally to avoid hitting the director CPU 4096 LUN limit before we hit the MV LUN limit, if both ports are available from a single VMAX director CPU (e.g. 7E-0 and 7E-1), one will be discarded and not used. In this way the planner only uses a one port per VMAX director CPU.

Next, ViPR Controller determines the number of ports available on the Network with the fewest number of usable ports. This will determine how many MVs are created, according to the following table (not all combinations are shown):

Table 3. MV Creation Based on Port Availability in Network with Least Usable Ports

Number of Networks	Number of Ports in Fewest Port Network	Number of MVs Created	Number of Ports in Each MV
1	1	0 (no redundancy)	1
1	2	1	2
1	4	2	2
1	8	4	2
2	1	1	2
2	2	1	4
2	4	2	4
2	8	2	8
2	12	4	6
2	16	4	8

Number of Networks	Number of Ports in Fewest Port Network	Number of MVs Created	Number of Ports in Each MV
2	24	6	8

How ViPR Controller Provisions the Automatically Generated VMAX Masking Views

ViPR Controller creates a Cascaded Initiator Group for each MV. It is constructed of two Initiator Groups that contain the physical Initiators from the VPLEX cluster being used. The screenshot below shows two Cascaded Initiator Groups (used by two different MVs) that are sharing the two physical Initiator Groups each containing four Initiators.

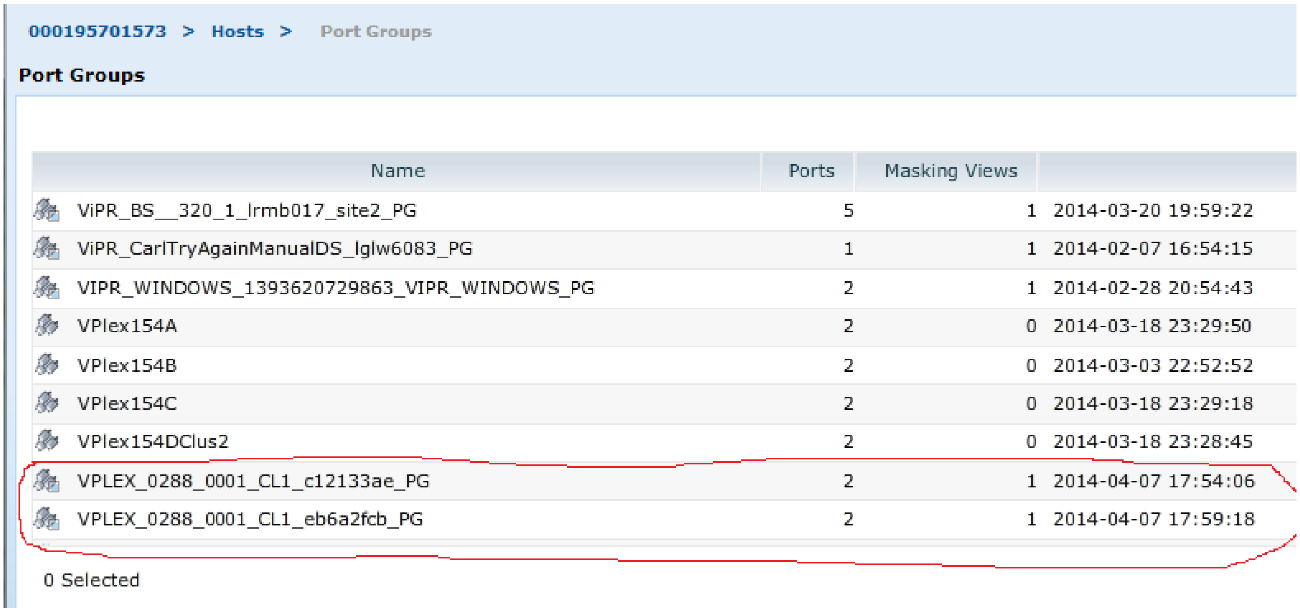
000195701573 > Hosts > Initiator Groups

Initiator Groups

Name	Parent IGs	Child IGs	Masking Views	Initiators	Consistent LUNs	Port Flag Override
ViPR_BS__320_1_lrmb017_site2_CIG	0	1	1	8	Yes	No
ViPR_CarlTryAgainManualDS_lglw6083_CIG	0	1	1	8	Yes	No
VIPR_WINDOWS_1393620729863_VIPR_WINDOWS_CIG	0	2	1	4	Yes	No
VmaxExp273961_sanityCluster1_host40aeip_10_247_66_206...	0	1	0	1	Yes	No
voljer2_lab_voyence_com_IG	7	0	0	1	Yes	No
voljer_lab_voyence_com_IG	7	0	0	1	Yes	No
VPLEX_0288_0001_CL1_c12133ae_CIG	0	2	1	8	No	No
VPLEX_0288_0001_CL1_eb6a2fcb_CIG	0	2	1	8	No	No
VPLEX_0288_0001_CL1_IG1_IG	2	0	2	4	No	No
VPLEX_0288_0001_CL1_IG2_IG	2	0	2	4	No	No

0 Selected

ViPR Controller creates a Port Group for each MV. It has the number of ports allocated that ViPR Controller determined was optimal. No ports in one group are used in another Port Group. Here are the Port Groups corresponding with the above example:



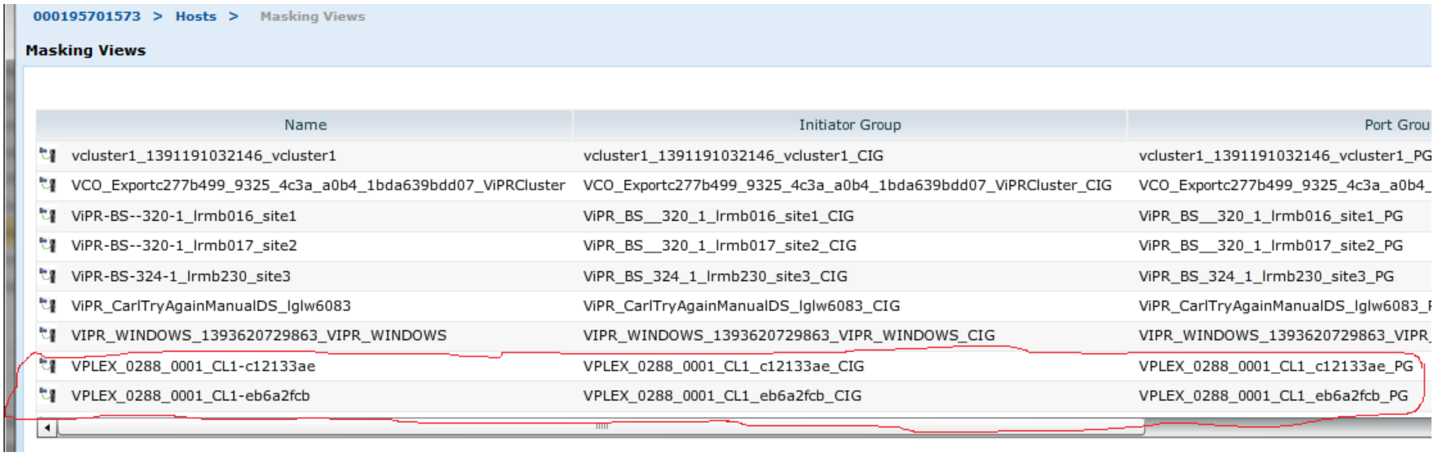
000195701573 > Hosts > Port Groups

Port Groups

Name	Ports	Masking Views	
ViPR_BS__320_1_lrmb017_site2_PG	5	1	2014-03-20 19:59:22
ViPR_CarlTryAgainManualDS_lglw6083_PG	1	1	2014-02-07 16:54:15
VIPR_WINDOWS_1393620729863_VIPR_WINDOWS_PG	2	1	2014-02-28 20:54:43
VPlex154A	2	0	2014-03-18 23:29:50
VPlex154B	2	0	2014-03-03 22:52:52
VPlex154C	2	0	2014-03-18 23:29:18
VPlex154DClus2	2	0	2014-03-18 23:28:45
VPLEX_0288_0001_CL1_c12133ae_PG	2	1	2014-04-07 17:54:06
VPLEX_0288_0001_CL1_eb6a2fcb_PG	2	1	2014-04-07 17:59:18

0 Selected

A MV is made from each of the Port Groups in combination with a Cascaded Initiator Group:

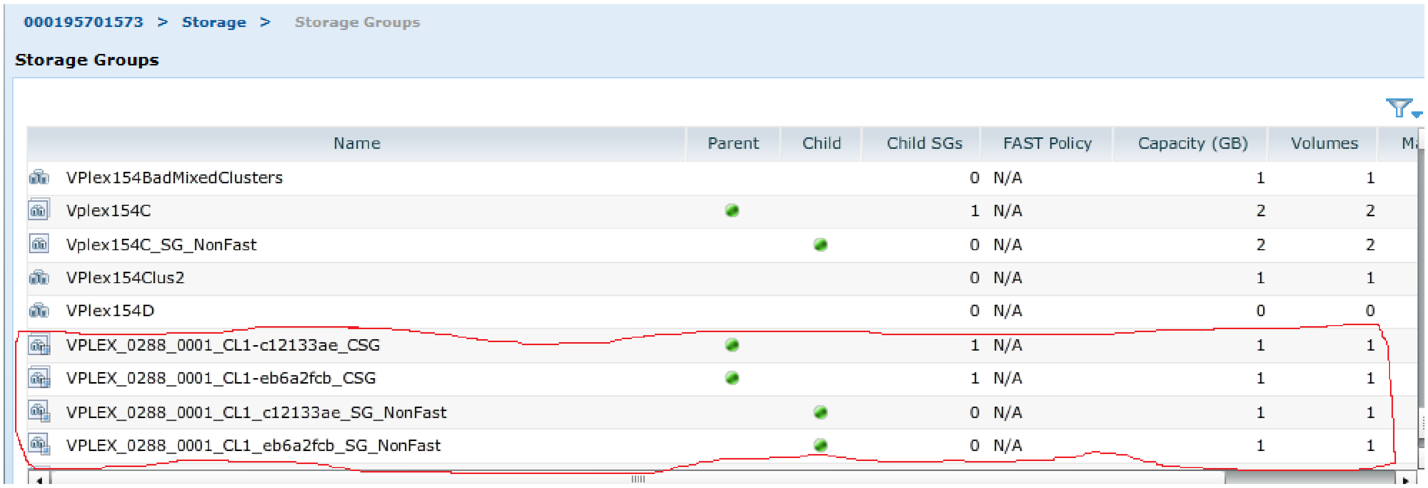


000195701573 > Hosts > Masking Views

Masking Views

Name	Initiator Group	Port Grou
vcluster1_1391191032146_vcluster1	vcluster1_1391191032146_vcluster1_CIG	vcluster1_1391191032146_vcluster1_PG
VCO_Exportc277b499_9325_4c3a_a0b4_1bda639bdd07_ViPRCluster	VCO_Exportc277b499_9325_4c3a_a0b4_1bda639bdd07_ViPRCluster_CIG	VCO_Exportc277b499_9325_4c3a_a0b4_
ViPR-BS--320-1_lrmb016_site1	ViPR_BS__320_1_lrmb016_site1_CIG	ViPR_BS__320_1_lrmb016_site1_PG
ViPR-BS--320-1_lrmb017_site2	ViPR_BS__320_1_lrmb017_site2_CIG	ViPR_BS__320_1_lrmb017_site2_PG
ViPR-BS-324-1_lrmb230_site3	ViPR_BS_324_1_lrmb230_site3_CIG	ViPR_BS_324_1_lrmb230_site3_PG
ViPR_CarlTryAgainManualDS_lglw6083	ViPR_CarlTryAgainManualDS_lglw6083_CIG	ViPR_CarlTryAgainManualDS_lglw6083_I
VIPR_WINDOWS_1393620729863_VIPR_WINDOWS	VIPR_WINDOWS_1393620729863_VIPR_WINDOWS_CIG	VIPR_WINDOWS_1393620729863_VIPR_
VPLEX_0288_0001_CL1-c12133ae	VPLEX_0288_0001_CL1_c12133ae_CIG	VPLEX_0288_0001_CL1_c12133ae_PG
VPLEX_0288_0001_CL1-eb6a2fcb	VPLEX_0288_0001_CL1_eb6a2fcb_CIG	VPLEX_0288_0001_CL1_eb6a2fcb_PG

Finally, a Cascaded Storage Group is associated with the MV. It holds one or more child Storage Groups (based on FAST policy) which hold the volumes:



000195701573 > Storage > Storage Groups

Storage Groups

Name	Parent	Child	Child SGs	FAST Policy	Capacity (GB)	Volumes	M
VPlex154BadMixedClusters			0	N/A	1	1	
Vplex154C	●		1	N/A	2	2	
Vplex154C_SG_NonFast		●	0	N/A	2	2	
VPlex154Clus2			0	N/A	1	1	
VPlex154D			0	N/A	0	0	
VPLEX_0288_0001_CL1-c12133ae_CSG	●		1	N/A	1	1	
VPLEX_0288_0001_CL1-eb6a2fcb_CSG	●		1	N/A	1	1	
VPLEX_0288_0001_CL1_c12133ae_SG_NonFast		●	0	N/A	1	1	
VPLEX_0288_0001_CL1_eb6a2fcb_SG_NonFast		●	0	N/A	1	1	

How ViPR Controller Generates Storage Groups for VPLEX Use on the VNX

ViPR Controller will also generate Storage Groups for the VNX automatically if no suitable Storage Groups are found to hold VPLEX volumes. The VNX generation is somewhat different than the VMAX generation, so it is useful to explain it here.

On the VNX, either one or two Storage Groups are generated, depending both on the Initiator (VPLEX back-end port) configuration and on the available VNX ports. First the algorithm determines if the Initiators can be partitioned into two Initiator Group sets. For this to occur, the following conditions must be true:

1. All four Initiators on each director *must* be operational, connected to the array, and usable by the Virtual Array.
2. The VPLEX back-end ports must be wired such that the even ports are on one Network, and the odd ports are on a second Network. This means if a director has four back-end ports are A1-FC00, A1-FC01, A1-FC02, and A1-FC03 then ports A1-FC00 and A1-FC02 should be wired to Network one, and ports A1-FC01 and A1-FC03 should be wired to Network two. Alternately, they can be wired to four separate Networks. This must be true for each VPLEX director.

If the above conditions are met, two Initiator Groups will be generated and each Initiator Group will be used as a host for a separate Storage Group. Otherwise all the Initiators will be combined into a single Storage Group.

In order to generate two Storage Groups, it is also necessary to partition the available ports into two Port Groups. For this to happen there must be at least two Networks, and each Network must contain at least one SP-A and one SP-B port.

If the administrator wants to manually configure the Storage Groups on the VNX for use by the VPLEX, this can also be done similarly to how it is done for the VMAX. The Storage Groups must pass the same validation logic as MVs. Manual setup of the Storage Groups is not discussed in this document.

Considerations for Configuring Your Virtual Array(s)

You should think about the implications that your Virtual Array configurations have on the allocation of Storage Ports from the arrays and how that will affect the MVs or Storage Groups that ViPR Controller generates.

Here are some considerations to think about:

- You may want to dedicate storage array ports for use by the VPLEX. This would avoid problems caused because users other than the VPLEX are counting against the VMAX director CPU LUN limit. The best way to do this would be to set up a separate Virtual Array for use exclusively by each VPLEX cluster, and to exclusively assign array ports to those Virtual Arrays.
- You may want to avoid use of ports sharing the same VMAX director CPU with a port being used for VPLEX (e.g. 7E-0 and 7E-1). The best way to ensure this is to make a Virtual Array that is not actually used (e.g. the *Unallocated Virtual Array*), and assign the ports you want to remain unused there.
- You should never mix Initiators from both VPLEX clusters in one Virtual Array. That is an invalid configuration that will fail to provision.
- You should assign all the VPLEX back-end ports for a given cluster to the Virtual Array(s) used for that cluster. There is no point in partitioning them among Virtual Arrays.

Setting Up the Pre-Defined Masking Views

If a user wants to pre-define multiple MVs on a VMAX, this must be done before using ViPR Controller to create a VPLEX virtual volume using the specified backend array. If creation of a virtual volume is attempted before the predefined MVs are in place, we know from the above algorithm that ViPR Controller will then automatically create MV(s) automatically, which will likely conflict with any future predefined MVs the administrator would subsequently create.

Note: Predefined MVs must be created before any attempt is made to create VPLEX volumes using a given array as the backing store.

Planning the VMAX Masking Views

You should carefully plan the layout of your VMAX MVs that are to be pre-created. The number of MVs that can be created depends on several factors, including:

- How many VPLEX back ports are on fabrics/VSANs that are connected to the VMAX
- How many VMAX ports are on those same fabrics
- How many director CPUs are used for the ports

- Redundancy considerations for the VMAX ports, i.e. we prefer ports on different directors or engines to be used together for a MV.

Note: Refer to the document Implementation and Planning Best Practices for EMC VPLEX for information on VPLEX best practices. Some of these practices are summarized here.

There are a few basic rules that *must* be satisfied for a viable VPLEX configuration, according to the VPLEX best practices:

- Every director must have at least two paths to all storage.
- No director should have more than four paths to any storage. Having more than four paths causes issues with timeouts taking too long before switching to alternate directors. This can cause connectivity loss.

Initiator Groups

The VPLEX back-end ports are used as Initiators to the VMAX. In this document, the term *initiators* refers to the VPLEX back end ports that are used for communication with the VMAX.

You must create an Initiator Group consisting of VPLEX initiators (VPLEX back-end ports) from one of the VPLEX clusters on the VMAX. The Initiator Group should consist of at least two initiators from each VPLEX director on either cluster-1 or cluster-2 (but not both clusters). Ideally, the initiators are split across two different networks for redundancy.

If all four back-end ports on every director in a VPLEX cluster can be connected to a specific VMAX, it is possible to split the initiators into two groups, one containing two ports from each VPLEX director, and the other containing the other two ports from each VPLEX director. Within each group, a VPLEX director's two ports should be on different networks so as to avoid failure caused by a network outage. However, in the example configurations below all the VPLEX initiators from one cluster are included in a single Initiator Group.

For each MV you want to create, you want to set up a separate Cascaded Initiator Group (parent) that includes as its only member the Initiator Group (child) containing the VPLEX initiators. This parent Cascaded Initiator Group is associated with the MV. The Initiator Group containing the VPLEX initiators should not be directly associated with any MVs. Following this strategy will allow each MV to have a separate HLU space of 4096 LUNs.

As an example, with four ports on the VMAX that can be connected to the VPLEX initiators, you could set up the following MVs:

Initiator Groups:

IG = { all ports from VPLEX cluster1, which on my system are:

50:00:14:42:60:7D:C4:10, 50:00:14:42:60:7D:C4:11, 50:00:14:42:60:7D:C4:12, 50:00:14:42:60:7D:C4:13,
50:00:14:42:70:7D:C4:10, 50:00:14:42:70:7D:C4:11, 50:00:14:42:70:7D:C4:12, 50:00:14:42:70:7D:C4:13 }

Cascaded Initiator Groups (each one contains IG):

CIG1 = { IG }
CIG2 = { IG }

… additional Cascaded Initiator Group parents could be created …

Note: When ViPR Controller automatically generates Initiator Groups for the VPLEX, it sets the `Consistent LUNs` flag to false. If you are mixing manually created MVs with ViPR Controller created MVs for the VPLEX, `Consistent LUNs` must be set to `false` or you will get a consistent LUN violation error.

Port Groups

For these examples, assume there are two Networks, NetA and NetB, and that all the even VPLEX and VMAX ports are on NetA, and all the odd VPLEX and VMAX ports are on NetB. (Many other valid configurations are possible. This is one example.)

With eight ports, you can up scalability by utilizing additional VMAX director CPUs:

PG2A = FA-7E0, FA-7E1, FA-10E0, FA-10E1 (four ports, redundant directors, two CPUs)
PG2B = FA-7F0, FA-7F1, FA-10F0, FA-10F1 (four ports, redundant directors, two additional CPUs)
MV1 = { CIG1, PG2A }
MV2 = { CIG2, PG2B}

Zoning becomes:

MV1: (notice only four paths per director are used)
A1-FC00 -> FA-7E0 (director 1-1-A)
A1-FC01-> FA-7E1
A1-FC02-> FA-10E0
A1-FC03-> FA-10E1
B1-FC00-> FA-7E0 (director 1-1-B, sees same ports as director 1-1-B)
B1-FC01-> FA-7E1
B1-FC02-> FA-10E0

B1-FC03-> FA10E1

MV2: (notice only four paths per director, and that all ports are separate from MV1):
A1-FC00 -> FA-7F0 (director 1-1-A)
A1-FC01-> FA-7F1
A1-FC02-> FA10F0
A1-FC03-> FA10F1
B1-FC00-> FA-7F0 (director 1-1-B, sees same ports as director 1-1-A)
B1-FC01-> FA-7F1
B1-FC02-> FA10F0
B1-FC03-> FA10F1

So with eight ports, each MV uses a disjoint set of director CPUs. MV1 uses the CPUs 7E and 10E, and MV2 uses the CPUs 7F and 10F. Now each MV can scale to 4096 (non-meta) volumes. Since volumes will be split evenly across the MVs, more total volumes (8192) can be supported.

With even more ports available, and four Cascading Initiator Groups, you can create four MVs, while still using a unique set of director CPUs for each MV. Consider 16 ports:

Port Groups (each has an independent set of CPUs):

PG4A = FA-7E0, FA-7E1, FA-10E0, FA-10E1
PG4B = FA-7F0, FA-7F1, FA-10F0, FA-10F1
PG4C = FA-9E0, FA-9E1, FA-8E0, FA-8E1
PG4D= FA-9F0, FA-9F1, FA-8F0, FA-8F1
MV1 = { CIG1, PG4A }
MV2 = { CIG2, PG4B}
MV3 = { CIG3, PG4C}
MV4 = { CIG4, PG4D}

If you double the number of ports again to 32, you could support eight MVs in a similar configuration. The following are some observations from these simple examples:

- You should use a separate Cascading Initiator Group parent for each MV. The child Initiator Group(s) (containing the VPLEX initiators) should not be directly associated with a MV.
- If you want four paths per director, you need a minimum of four ports in each MV. This assumes that all VPLEX directors share the same four ports. This says an upper bound on the number of MVs is the number of ports divided by four,

assuming they are evenly split across networks and each VMAX CPU has one port connected to each network.

- You could potentially get more overall bandwidth from a MV by assigning more than four ports per MV, but each director can only use a maximum of four ports.
- You can use the two ports from a single VMAX CPU on different networks within the same MV without suffering any scalability loss.

A Simple Example Setup on a VMAX

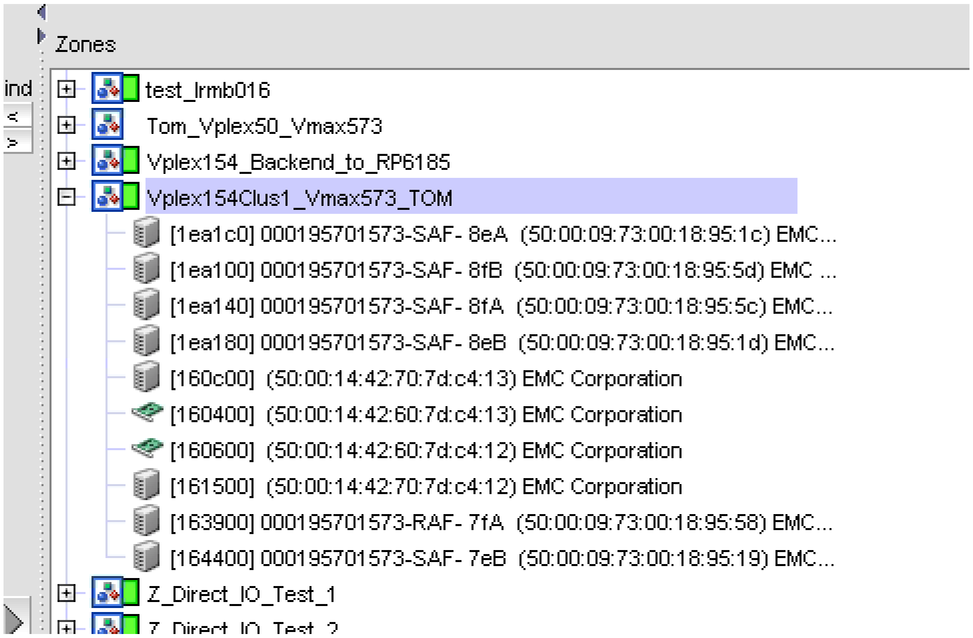
This section shows how to set up a simple multiple MV configuration on a VMAX. Instructions are provided in a Step-by-Step sequence. You must use values and configuration appropriate to the specific VPLEX and VMAX you are configuring.

If you have not planned your configuration of Port Groups, Initiator Groups, and MVs, do so before proceeding.

Note: In a metro VPLEX, you should provision MVs for each cluster as a separate operation.

Step 1: Configure the Zoning

In the simple configuration described in this section, there are two VPLEX directors in a VPLEX cluster. Each director has two back-end ports connected to the backing array. There is only a single Network. There are six ports on the backing array in that Network. There is one zone containing the four VPLEX back-end ports and the six array ports:



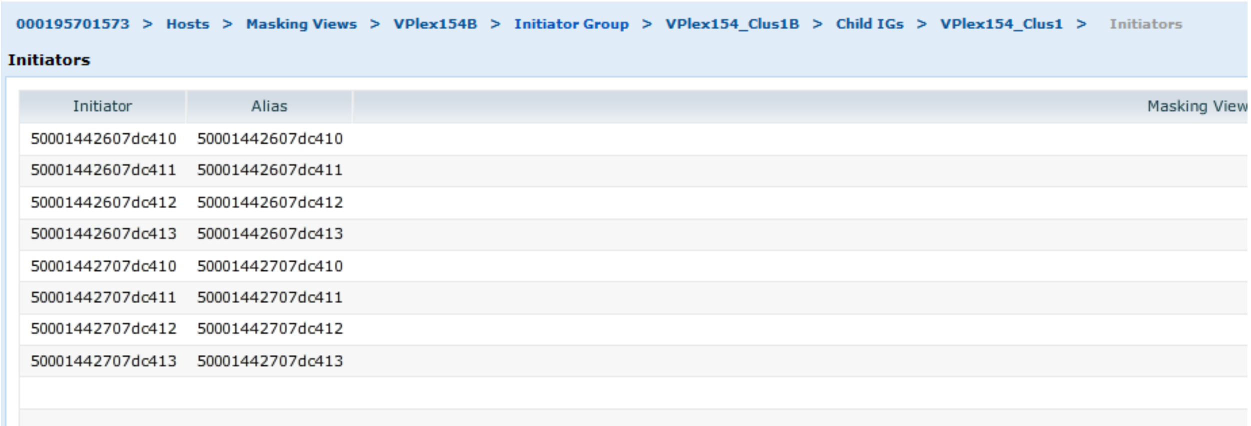
Notes: You must not mix Initiators from VPLEX Cluster-1 and VPLEX Cluster-2 in the same zone. Create separate zones for the Initiators from each Cluster.

In production use, the VPLEX Initiators should be split across two different Networks. Each network would then be zoned separately.

Zoning must be successfully completed for the VMAX to *see* the Initiators on the fabric, allowing you to easily create the Initiator Groups in Unisphere.

Step 2: Configure the Initiator Groups

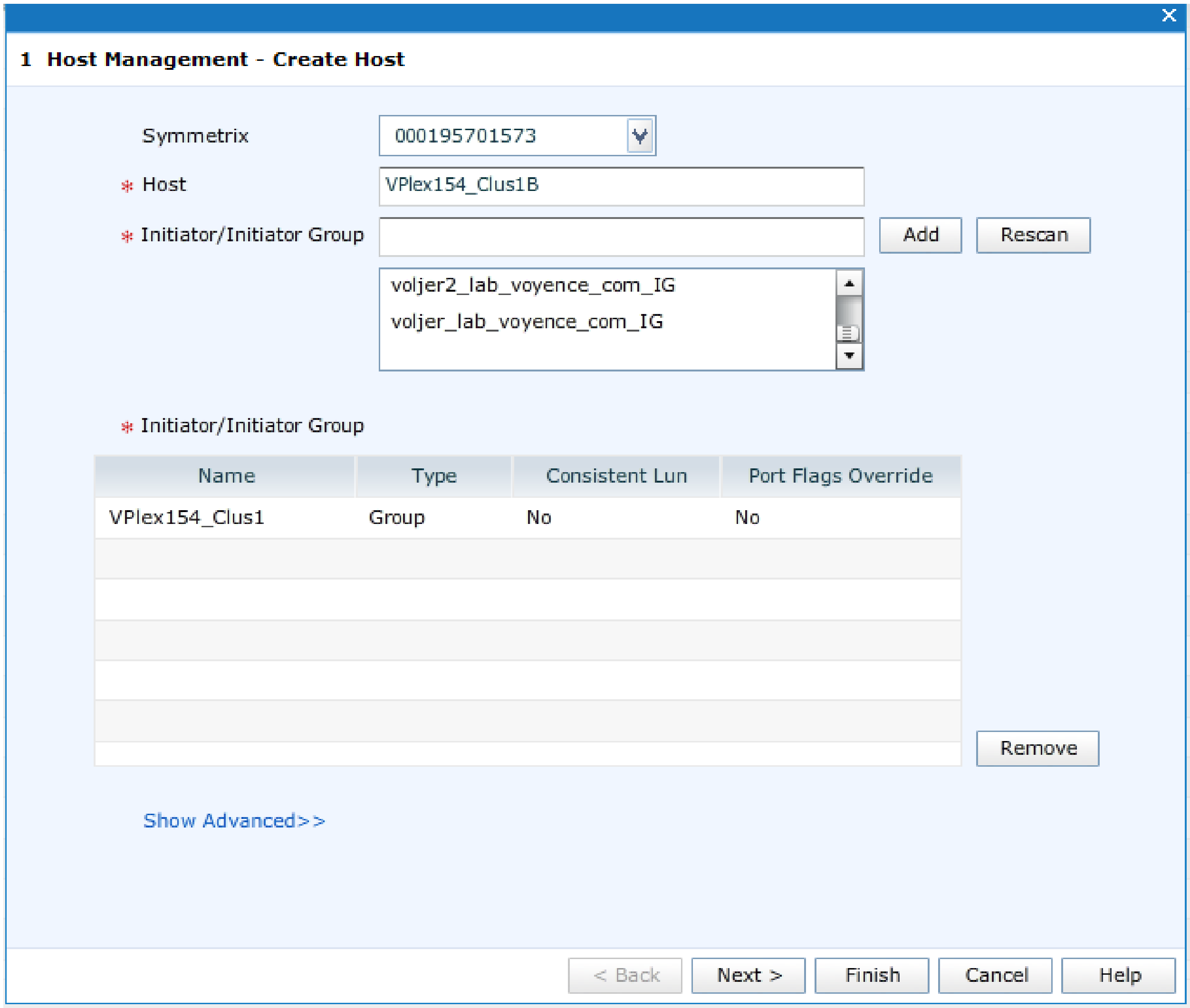
Configure Initiator Groups for the VPLEX back-end ports. In this example configuration, there are four ports per VPLEX director connected to the array (using two Networks). Only one Initiator Group is built.



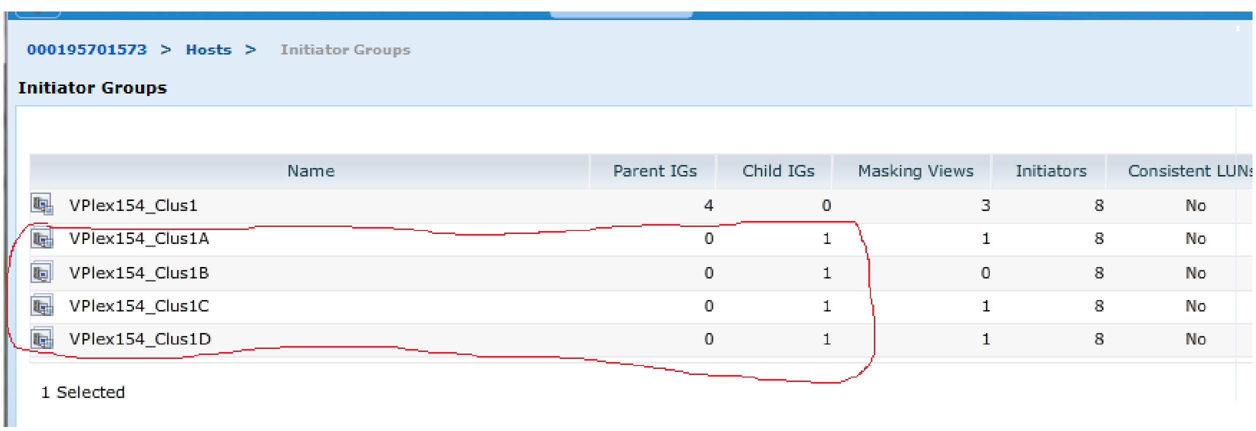
Initiator	Alias	Masking View
50001442607dc410	50001442607dc410	
50001442607dc411	50001442607dc411	
50001442607dc412	50001442607dc412	
50001442607dc413	50001442607dc413	
50001442707dc410	50001442707dc410	
50001442707dc411	50001442707dc411	
50001442707dc412	50001442707dc412	
50001442707dc413	50001442707dc413	

Step 3: Create the Cascaded Initiator Groups

For each MV to be created, create a Cascaded Initiator Group parent that holds the above Initiator Group:



Repeat as necessary to have a Cascaded Initiator Group for each MV:



Step 4: Configure the Port Groups

This example configuration has a *very* limited number of ports. The administrator can only afford two ports per MV using the six available ports. This does not allow for an ideal level of redundancy.

Create three Port Groups:

000195701573 > Hosts > Port Groups > VPlex154A >

Ports

Dir:Port	Port Groups	Masking Views
FA-8E:0	5	5
FA-8E:1	5	5

000195701573 > Hosts > Port Groups > VPlex154B > Ports

Ports

Dir:Port	Port Groups	Masking Views
FA-8F:0	3	3
FA-8F:1	3	4

000195701573 > Hosts > Port Groups > VPlex154C > Ports

Ports

Dir:Port	Port Groups	Masking Views
FA-7E:1	8	8
FA-8F:1	3	4

Note: Two of the Port Groups, VPLEX 154B and VPLEX 154C, share a port FA-8F:1. This is not recommended and not a correct configuration.

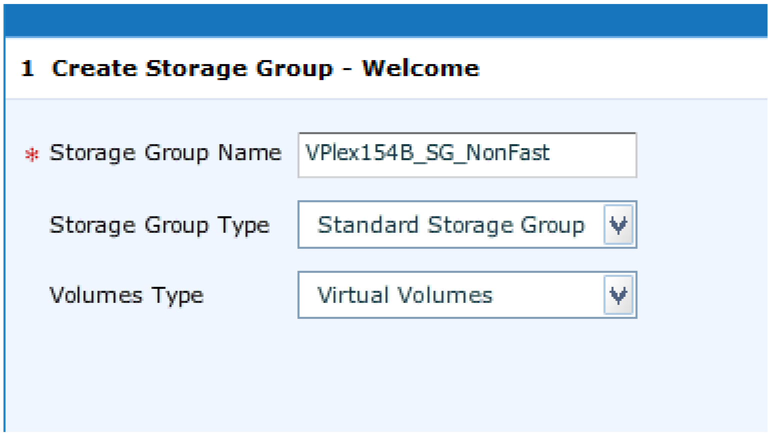
Step 5: Configure the Cascading Storage Groups

This step is repeated once for each MV. This procedure shows the creation steps necessary for one MV.

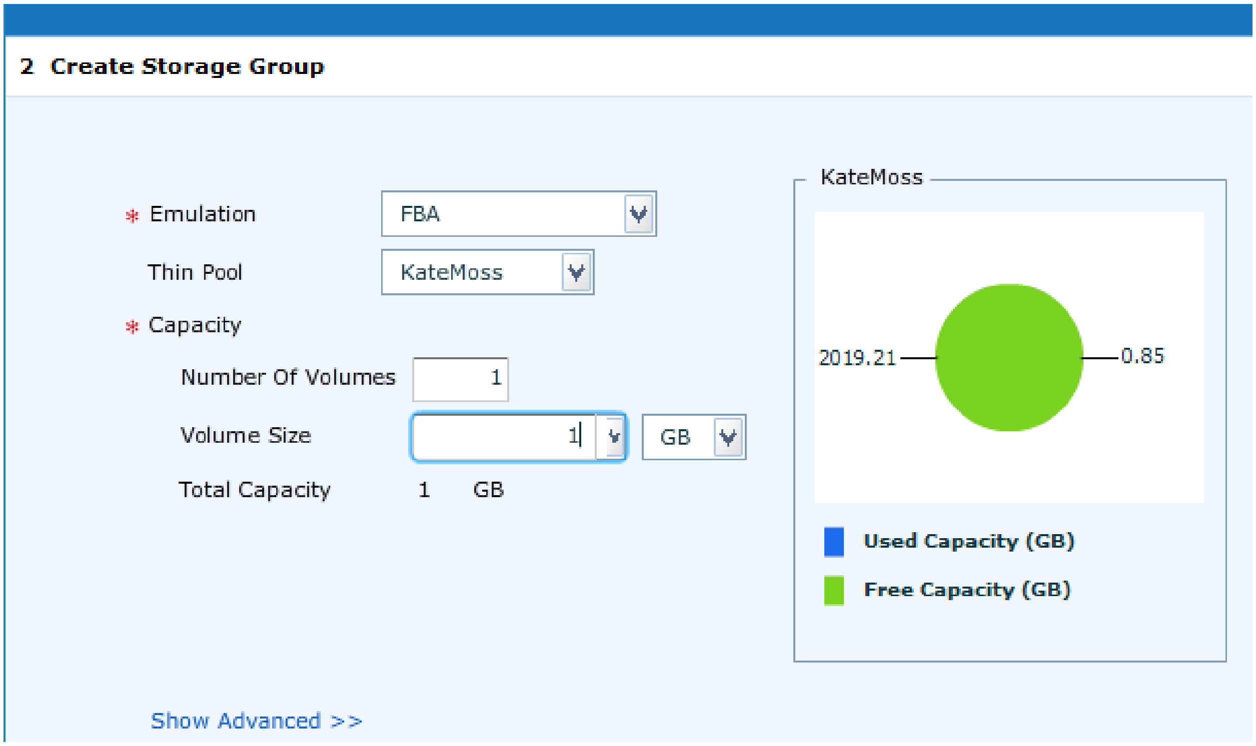
Step 5a: Create the Initial Storage Group (will be a Child Storage Group)

This step creates a child Storage Group with an unused, arbitrary volume. (Unisphere does not allow you to create a Storage Group without a volume.)

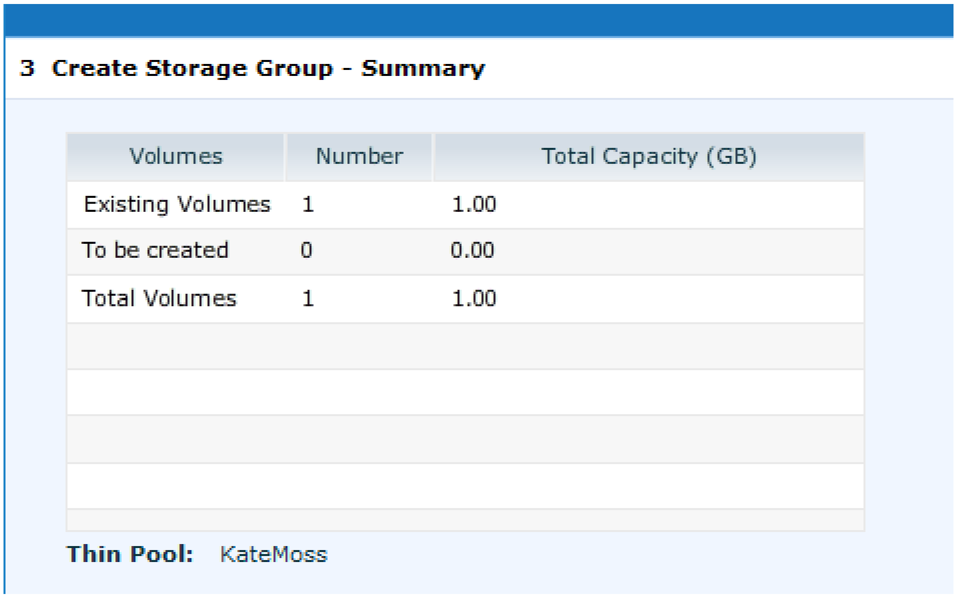
Note: The name for this Storage Group *must* end in `_SG_NonFast`:



Create an arbitrarily small (unused) volume for this Storage Group:



Click **Finish** to complete the request.

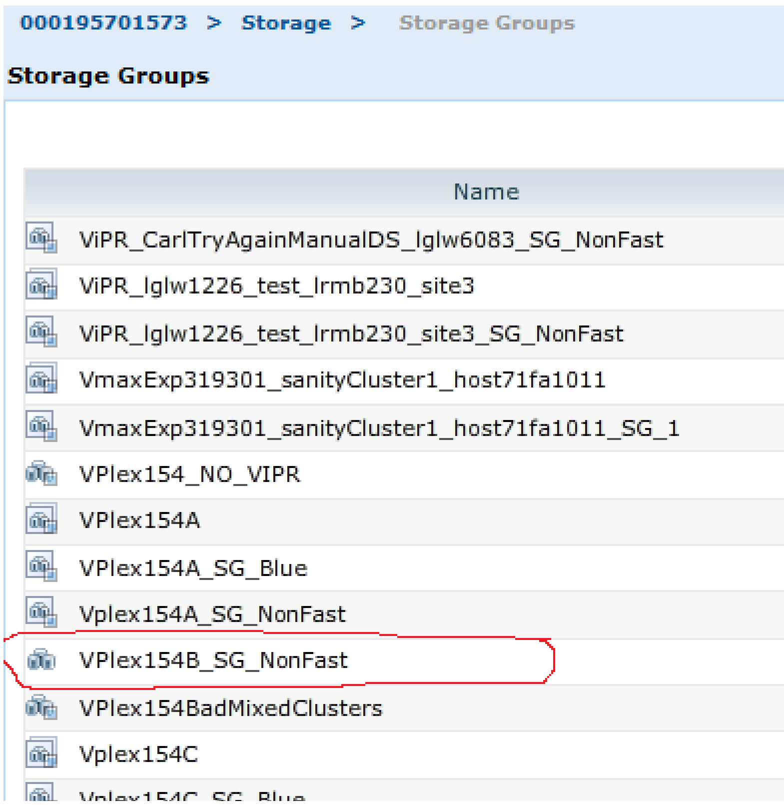


3 Create Storage Group - Summary

Volumes	Number	Total Capacity (GB)
Existing Volumes	1	1.00
To be created	0	0.00
Total Volumes	1	1.00

Thin Pool: KateMoss

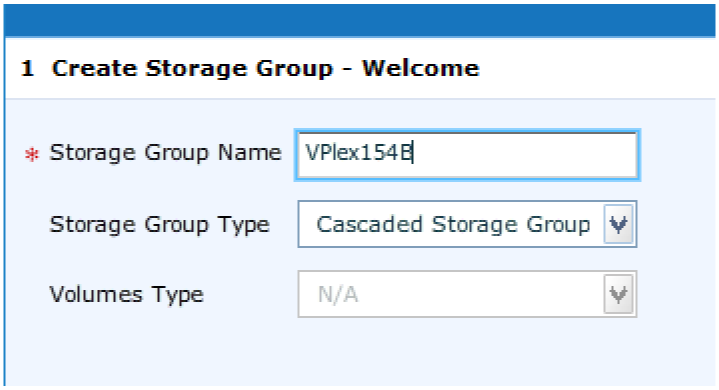
Verify that the new Storage Group is created:



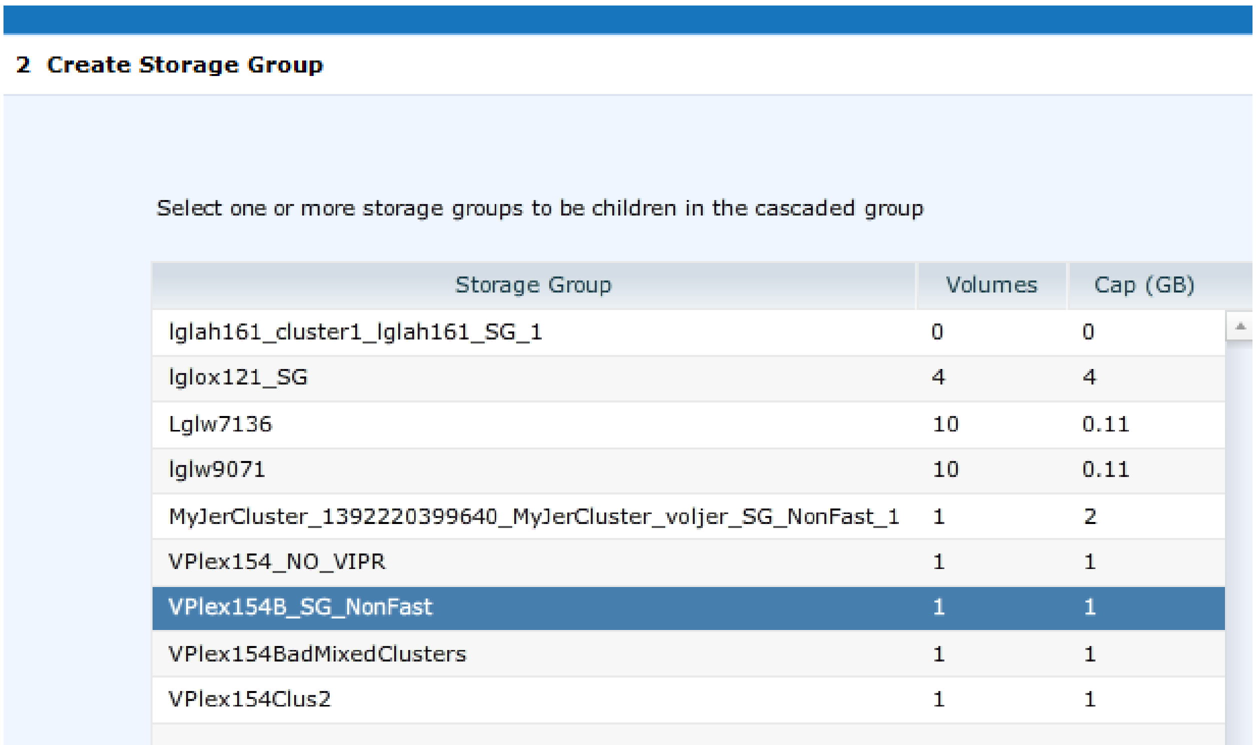
Step 5b: Create the Cascaded Storage Group (Parent)

Now create a cascading Storage Group to hold the previously created Storage Group which becomes the child:

Note: The name for this Storage Group *must* be the same as the previous one without the `_SG_NonFast` suffix.

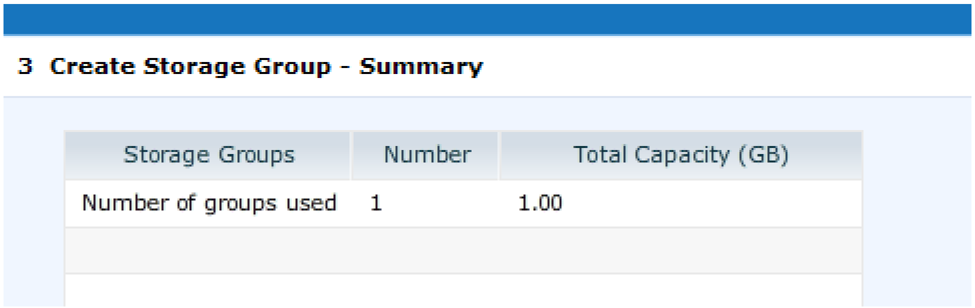


Select the previously created Storage Group as the child:



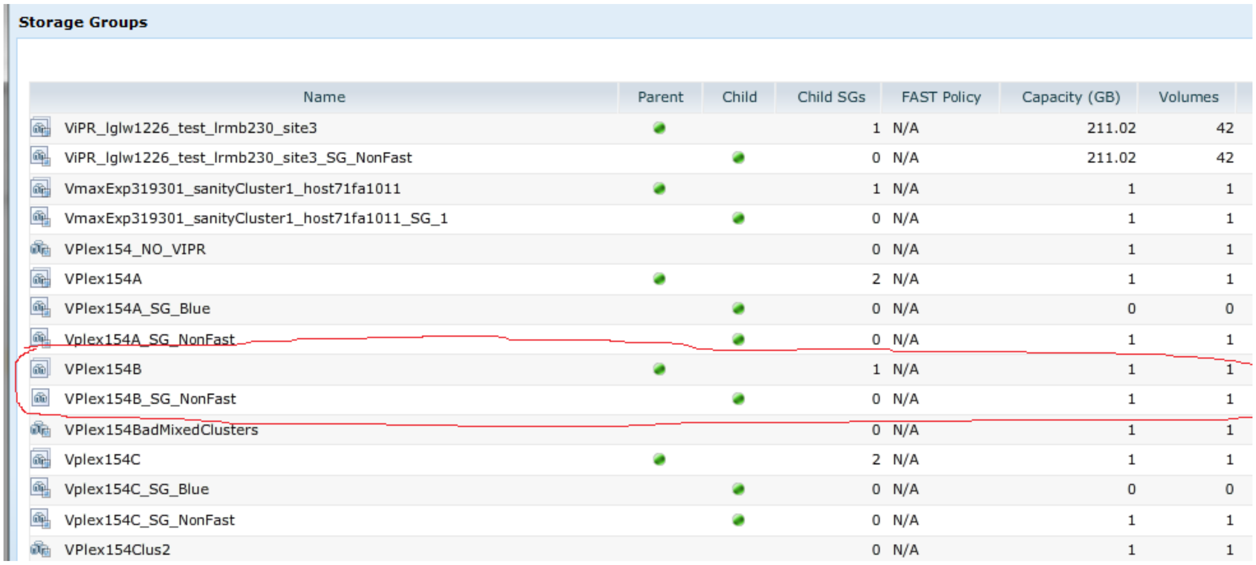
Storage Group	Volumes	Cap (GB)
lglah161_cluster1_lglah161_SG_1	0	0
lglox121_SG	4	4
Lglw7136	10	0.11
lglw9071	10	0.11
MyJerCluster_1392220399640_MyJerCluster_voljer_SG_NonFast_1	1	2
VPlex154_NO_VIPR	1	1
VPlex154B_SG_NonFast	1	1
VPlex154BadMixedClusters	1	1
VPlex154Clus2	1	1

Click **Finish** on this screen:



Storage Groups	Number	Total Capacity (GB)
Number of groups used	1	1.00

Verify the configuration of the cascaded Storage Groups:



Storage Groups

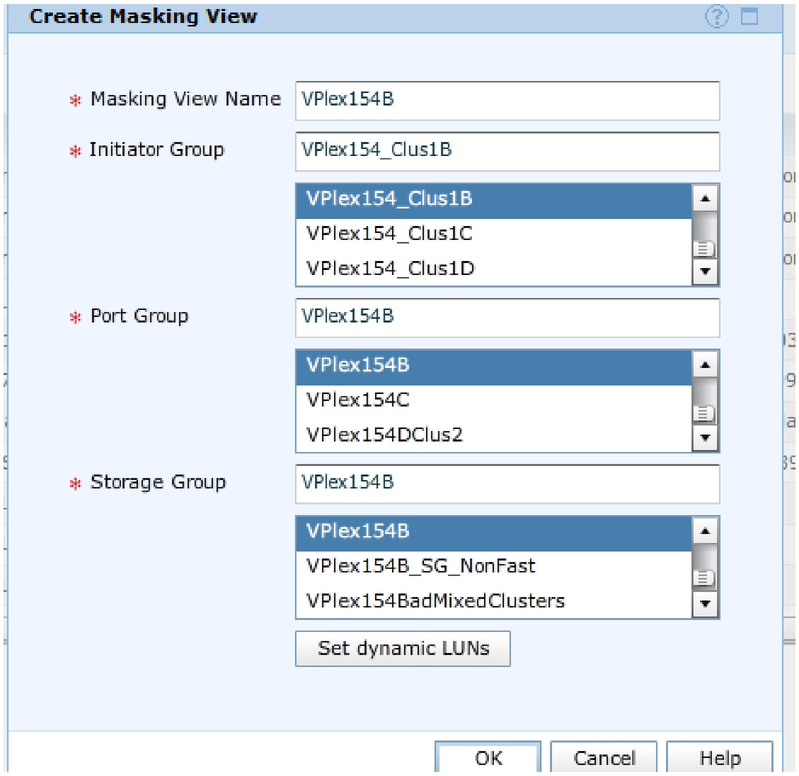
Name	Parent	Child	Child SGs	FAST Policy	Capacity (GB)	Volumes
ViPR_lglw1226_test_lrmb230_site3	●		1	N/A	211.02	42
ViPR_lglw1226_test_lrmb230_site3_SG_NonFast		●	0	N/A	211.02	42
VmaxExp319301_sanityCluster1_host71fa1011	●		1	N/A	1	1
VmaxExp319301_sanityCluster1_host71fa1011_SG_1		●	0	N/A	1	1
VPlex154_NO_VIPR			0	N/A	1	1
VPlex154A	●		2	N/A	1	1
VPlex154A_SG_Blue		●	0	N/A	0	0
Vplex154A_SG_NonFast		●	0	N/A	1	1
VPlex154B	●		1	N/A	1	1
VPlex154B_SG_NonFast		●	0	N/A	1	1
VPlex154BadMixedClusters			0	N/A	1	1
Vplex154C	●		2	N/A	1	1
Vplex154C_SG_Blue		●	0	N/A	0	0
Vplex154C_SG_NonFast		●	0	N/A	1	1
VPlex154Clus2			0	N/A	1	1

Repeat Step 5 as necessary so as to create a pair of cascaded Storage Groups for each MV that is planned.

Step 6: Create Masking Views

Using the components you have created, you may now assemble the MVs. This example shows creating a single MV for the Cascaded Storage Groups created above. You should repeat Step 4 for each MV.

Create the MV using the Parent Storage Group, with the correct Initiator Group and Port Group:



Validating Your Export Masks

You cannot be certain your manually created Export Masks are valid until you create at least as many volumes using ViPR Controller as the number of Export Masks. ViPR Controller should round-robin the assignment of volumes to Export Masks.

Note: Look in the `controllersvc.log` as shown above and confirm that each Export Mask you created is passing ViPR Controller's validation checks.

Special Situations

This section describes any special situations that might arise in the design or provisioning of Export Masks on the VMAX for use by a VPLEX.

Cross-Connected FC Networks Between the VPLEX Back-End Ports and the VMAX Array(s)

The only reason to cross-connect VPLEX back-end ports to VMAX arrays at different sites (using different clusters) is so that in situations with very few arrays, the VPLEX metadata and logging volumes can be protected using mirroring.

In this configuration, zoning and masking are set up so that Initiators from both VPLEX clusters can access targets on the array(s). ViPR Controller does not support this configuration for provisioning and *must not* use Export Masks so configured. There are two ways this should be prevented:

- ViPR Controller does an explicit validation check that MVs with Initiators from both clusters of a VPLEX should not be used.
- The administrator *should* include `NO_VIPR` or `no_vipr` in the Export Mask name so that ViPR Controller will not attempt to use it and it is clear that it was not intended for ViPR Controller.

Note: Under no circumstances should regular backing volumes for VPLEX virtual volumes be put in such a MV. This will cause provisioning errors.