Object Matrix

MatrixStore Administration and Programming Guidelines

Version 1.6, September 2018

Contents

1 About this guide

Every software solution has been designed with certain best usage practices in mind and MatrixStore is no exception to that rule. This guide is intended to provide advanced information on how to get the most out of a MatrixStore solution both from an administration point of view and from a programmer (using the MatrixStore API) point of view. Furthermore, the guide points to the future direction of MatrixStore and how to program / use MatrixStore in a way that will best take advantage of those future versions.

It is assumed that the reader of this guide is familiar with the MatrixStore Explained document, and if programming against MatrixStore, with the MatrixStore API documents.

2 MatrixStore Concepts

2.1 Maximum Cluster Size

The cluster is designed with an upper limit of 200 nodes in mind – however, in fact, under most circumstances the cluster could go far higher than 200 nodes.

- Object Matrix has tested clusters up to 46 nodes

2.2 Maximum Node Disk Size

Nodes of up to 624TB in size are expected for, however, whilst there is no actual node size limit in practice, there is an issue that the larger the node size, then in disaster situations where a node is lost, then the rest of the cluster will take longer to recover the data on the lost node. The time taken to recover data is approximately 400MB/s per node with data therefore imagine a cluster with the following configuration:

- 3 nodes
- 24TB per node in RAID6
- one node goes down and a new node is added

MTTR (mean time to recovery) would therefore be

- Data to recover = 20TB
- 2 nodes recovering data = 800MB/s
- MTTR = approx 7 hours.

Now imagine the same scenario but with 7 nodes and 1 going down. In that situation:

- Data to recover = 20TB

- 6 nodes recovering data = 2.4GB/s
- MTTR = approx 2 hours

As can be seen: As the number of nodes goes up MTTR goes down, whilst as the size of a node goes up MTTR goes up. Secondly, as the size of a node goes up in a disaster scenario the disk space required to recover data from that node also goes up. The limitation on the size of a node is purely down to having manageable chunks of data to handle in disaster scenarios.

Lastly, where write and read speed are a factor then the higher bandwidth given by larger numbers of nodes are an important factor to take into account:

- As of v3.2 cluster speed (rule of thumb) where multiple numbers of clients are writing stands at approx. 80MB/s per node in the cluster via a 1 GigE interface, 600MB/s per node via a 10 GigE interface.

2.3 Vaults

Vaults are described in MatrixStore Explained document.

- MatrixStore supports up to 2000 vaults per cluster
- [version 3.0 and above] supports up to 5000 vaults per cluster

Considerations to take into account when selecting a number of vaults are:

- Vaults cannot be deleted (in order to preserve audit trails) therefore incrementing the number of vaults over time above 2000 may become an issue
- The administration tool does not handle larger numbers of vaults elegantly – handling more than 100 vaults may become cumbersome

For those reasons it is suggested that a “Vault per User” in most organisations is not the optimal set up, and rather a “Vault per Department” or a “Vault per Project” (etc) type setup is usually a better approach.

2.4 Vault Users

Vault Users are described in MatrixStore Explained document.

- MatrixStore supports any number of users per Vault
- Each User gets its own log in credentials, and can be set with its own read, write, delete, search and attribute update permissions

The main reason for maintaining two users with the same permissions on a vault is for audit purposes: to be able to see what one user did on the vault, however, unlike in a normal filesystem, one user cannot be given individual permissions to access (e.g., to have read access) on individual objects in the vault.

2.5 Table of Cluster Capacity Guidelines

Type	Max	Notes
Nodes	40, 100, (200)	See max. cluster size
Objects	40,000,000 per node	See metadata
Metadata items	Typically 60,000,000 per node	Probably more of a restriction than the number of objects
Vaults	2000 (5000)	
Vault Users	No max.	If contemplating more than 100 then discuss with Object Matrix team, but theoretically the number could go to 1000s
Write speed	**1 GigE interfaces** - 100MB/sec a single file - 80MB/s per node in average **10 GigE interfaces** - 1000MB/sec a single file - 600MB/sec per node in average	Subject to the following limitations: - Native API is limited to 80MB/s for a single file - each client will be limited by its network bandwidth - any one file will not read/write at greater than interface speed

2.6 Vault User Credential Files

When a vault is created, or a user is added to a vault (in the MatrixStore Admin application – see the MatrixStore Admin Users Guide), the administrator is asked for a file location to store the details for vault access. These details include:

- Cluster ID: the GUID of the cluster being attached to
- User ID: the GUID of the user being attached to
- Vault ID: the GUID of the vault being attached to

- User Password: the password (or GUID) of the user being attached to
- IP addresses: used to locate at least one of the nodes of the cluster. Usually at least two should be given for failover, however, once attached the client will automatically discover the other nodes
- Communication Protocol: this will be one of:
    - rpc – synchronous non-encrypted connections
    - arpc – asynchronous non-encrypted connections
    - srpc – encrypted transmission of packets
    - sarpc – encrypted asynchronous connections

It is optionally, according to the organisation, where to keep these details, which are in clear text and therefore shouldn't be left somewhere insecure.

- Once loaded into a client application such as DropSpot or MXFS the details are kept on the local machine encrypted
- The original file could then be destroyed
- So long as the administrator has the overall system password all other passwords can be reset
- Object Matrix will generally keep a copy of the initial overall system password, however the administrator has the option to reset that password in the administration tool
- If the system administration password is completely lost an authorised Object Matrix engineer can regain that password on-site
- The customer has the option to be able to block that Object Matrix authorised engineer capability (at the risk of permanently losing all access to administration) if required

2.7 Certificates and LDAP Integration

The security model on the server includes the following characteristics:

- When connection is made to the server using a users credentials, and is accepted by the server, an access certificate is issued that allows the requested action to be performed within a reasonable time period
- All access credentials are sent within a SSL type transaction
- If the requested link transmission protocol is encrypted all following communications are sent encrypted, using a morphing encryption algorithm to thwart sniffing attacks

3 MatrixStore API Programming

3.1 DropSpot and MXFS Metadata Compatibility

Objects dropped into MatrixStore are not required to contain any metadata whatsoever, but in order for the objects to be viewed within DropSpot and/or MXFS certain metadata items must be added to the objects. The following table describes the attributes that are shared between MatrixStore applications and can be shared with thirdparty applications.

***M**/Mandatory: *=Mandatory, RO=Read Only*
***T**/Type: S=String, I=Integer, B=Boolean, L=Long*
***S**/Searchable: *=Yes.*

Name	M	T	S	Notes
MXFS_CREATION_TIME		L	*	Epoch, number of milliseconds since UTC Jan 01 1970, indicating when file has been created. Dropspot gets that attribute from the source file during copying it to the cluster. MXFS on MacOSX and Linux sets that value to a current time during archive operation. MXFS on Windows has that value explicitly set by OS to original timestamp of the source file. Application should not change that value later on. If application modifies this attribute it should also modify MXFS_CREATIONYEAR, MXFS_CREATIONMONTH and MXFS_CREATIONDAY to appropriate values
MXFS_CREATIONYEAR		I	*	Year extracted from MXFS_CREATION_TIME e.g. 2013. It should be updated whenever application modifies MXFS_CREATION_TIME
MXFS_CREATIONMONTH		I	*	Month extracted from MXFS_CREATION_TIME. Values from 1 (Jan) to 12 (Dec). It should be updated whenever application modifies MXFS_CREATION_TIME
MXFS_CREATIONDAY		I	*	Day extracted from MXFS_CREATION_TIME. Values from 1 to 31. It should be updated whenever application modifies MXFS_CREATION_TIME

Name	M	T	S	Notes
MXFS_ARCHIVE_TIME	RO	L	*	Generated by the cluster (v3.0+) and should not be changed. Epoch, number of milliseconds since UTC Jan 01 1970, indicating when file has been archived to the cluster. Application should not change that value later on.
MXFS_ARCHYEAR	RO	I	*	Generated by the cluster (v3.0+) and should not be changed. Year extracted from MXFS_ARCHIVE_TIME. Essential for compatibility with searching by date in DropSpot. E.g., 2011
MXFS_ARCHMONTH	RO	I	*	Generated by the cluster (v3.0+) and should not be changed. Month extracted from MXFS_ARCHIVE_TIME. Essential for compatibility with searching by date in DropSpot. From 1 (=Jan) to 12 (=Dec).
MXFS_ARCHDAY	RO	I	*	Generated by the cluster and should not be changed. Day extracted from MXFS_ARCHIVE_TIME. Essential for compatibility with searching by date in DropSpot. From 1 to 31.
MXFS_MODIFICATION_TIME		L	*	Epoch, number of milliseconds since UTC Jan 01 1970, indicating when was the last modification of the content (not metadata) of an object.
MXFS_ACCESS_TIME		L	*	Epoch, number of milliseconds since UTC Jan 01 1970, indicating when was the last time content (not metadata) of the object was read or written
DPSP_WEEK_OF_YEAR		I	*	Not used.
MXFS_TIMESTAMP				Deprecated. Should not be used. Please use MXFS_CREATION_TIME, MXFS_MODIFICATION_TIME, MXFS_ACCESS_TIME instead.
DPSP_TIMESTAMP		L	*	Deprecated. See MXFS_TIMESTAMP

Name	M	T	S	Notes
MXFS_PARENTOID	*	S	*	The MatrixStore object ID for the parent of this object. If this is a “top level” object then the parent OID should be set to be an empty string.
MXFS_FILENAME	*	S	*	Filename of the object, e.g., “movie.mpg”
MXFS_FILEEXT		S	*	Filename extension if known. E.g., “Doc”, “XLS” etc. Aids searching and sorting in DropSpot.
MXFS_FILENAME_UPPER	*	S	*	Uppercase version of MXFS_FILENAME
MXFS_INTRASH	*	B	*	This value must be set. Unless the file is to be put in the trashcan the value should be false.
MXFS_CATEGORY	*	I	*	Helps DropSpot identify the kind of contents the object contains. For correct behaviour, it should always be set to be a Folder when known. Categories are: *Unknown* = 0; *Folder* = 1; *Movies* = 2; Video, AVI, MPEG *Music* = 3; Audio, MP3 *Documents* = 4; MS Office, PDF, Text *Images* = 5; Images, JPG, GIF… *Vaults* = 6; MatrixStore Vaults
MXFS_DESCRIPTION		S	*	A string containing the published attributes pertaining to the object being stored. Generally, this value does not need to be set.
MXFS_GENERATOR		S	*	Set a code representing the writing application. E.g.“MXFS”
MXFS_LINK_SRC		S	*	Where an object is a link file to another object, this should be set to the (real) object’s MatrixStore ID.

Name	M	T	S	Notes
MXFS_MIMETYPE		S	*	String mimetype – helps DropSpot “double click open” files correctly. Examples are: "application/octet-stream" (Default) "text/directory" (Folders) "application/msword", "application/pdf", "audio/mp3", "application/x-wav", "audio", "video", "image", "text"
MXFS_OWNERGID		I	*	Non essential. If known set to the ID of the group the file belongs to.
MXFS_PATH	*	S	*	The absolute path of the original object. This is mandatory to be compatible with S3Connect and is the equivalent of your Amazon S3 object key.
MXFS_SIZE	RO	L	*	Used to retrieve the size of the object.
DPSP_TIMEBASEDUID		S	*	Important: A GUID that is entered into the metadata when an object is being stored. This way, if communication is lost with the server at a point in between the object write stream being closed, and receiving back the object ID for the object being stored, then on reconnection with the server the application storing the object can search for the object based upon the timebaseduid to see if in fact the object did finish writing. This can avoid duplicate writing of objects.
DPSP_DPSPUSEROID		S	*	The application user login name.
MXFS_USERNAME		S	*	In DropSpot this is the operating system login username. Where possible the original file user name should be preserved.
MXFS_COMPATIBLE		I	*	Integer denoting version number. Optional field currently set to “1” in DropSpot.
MXFS_ENCRYPTED		B		Not used.

3.2 Additional information regarding timestamp attributes

Since MatrixStore v3, on every object creation cluster generates automatically the following attributes:

- MXFS_CREATION_TIME, MXFS_MODIFICATION_TIME, MXFS_ACCESS_TIME, MXFS_ARCHIVE_TIME,
- MXFS_CREATIONYEAR, MXFS_CREATIONMONTH, MXFS_CREATIONDAY
- MXFS_ARCHYEAR, MXFS_ARCHMONTH, MXFS_ARCHDAY,

Attribute modification

- MXFS_CREATION_TIME, MXFS_MODIFICATION_TIME, MXFS_ACCESS_TIME can be changed by application and should be taken from timestamps of a source file if possible (Dropspot). MXFS is platform specific and most of the time only mtime of the source file (MXFS_MODIFICATION_TIME) will be preserved
- When application or API changes MXFS_CREATION_TIME it should also provide additional 3 MXFS_CREATIONxxx attributes for consistency
- MXFS_MODIFICATION_TIME, MXFS_ACCESS_TIME can be modified freely but only if application knows the details about source file, otherwise changing is not necessary
- MXFS_ARCHIVE_TIME, MXFS_ARCHYEAR, MXFS_ARCHMONTH, MXFS_ARCHDAY are read only. Applications should not try to update, remove them etc. In that scenario USUPPORTEDOPERATION error will be thrown. Also creating object with any of those attributes is blocked. API and applications (Dropspot etc) should comply. Those attributes will not be changed by the cluster.

3.3 Searchable keywords in DropSpot

From MatrixStore v2.4, all stand-alone keywords that are required to be searchable from DropSpot need to be entered with the keyword preceded by the string “_fs_”, therefore:
key="_fs_" + word you want to search. Keys should always be in lower case.
value=""

As an example: Superman movie.

File:
superman.mov

Sinopsis:
A strong man

If you want DropSpot and MXFS to see the file, add the mandatory attributes in the table and any others that are known.

- If you want DropSpot to find the file searching for simple string

"superman", add he attribute:
key="_fs_superman"; value=""

- If you want DropSpot to find the file searching for simple string
"mov", add he attribute:
key="_fs_mov"; value=""

- If you want DropSpot to find the file searching for simple string
"movie", add he attribute:
key="_fs_movie"; value=""

- If you want DropSpot to find the file searching for the string
"category=action", add he attribute:
key="category"; value="action"

With the above attributes, if you search for "movie superman" in DropSpot, it will construct an AND term searching for 2 attributes:
_fs_superman="" and _fs_movie="", finding the file as well.

As a minimum, the following attributes could be added to the file from the table:
MXFS_PARENTOID=string("")
MXFS_INTRASH=boolean(false)
MXFS_FILENAME=string("superman.mov")
MXFS_FILENAME_UPPER=string("SUPERMAN.MOV")
MXFS_CATEGORY=int(0)
MXFS_PATH=string("/superman.mov")

However, then as many other attributes as possible should be added.

4 System Metadata Attributes

MatrixStore provides access to system attributes in READ ONLY mode providing information to applications. The list is described below:

Name	Notes
__mxs__locked	Returns if the object is locked
__mxs__inCompliantStore*	Returns if the vault has regulation compliance switched on
__mxs__storesCurrentRetentionPeriod*	Returns the period of time an object will be locked from deletion / updating for.
__mxs__clustersCurrentTime*	Returns the current time in the cluster
__mxs__creationTime	Returns the creation time of the object. Returned as an 8 byte long.
__mxs__modifiedTime	Returns the last modified time of the object. Returned as an 8 byte long.
__mxs__length	Returns the length of the object. Returned as an 8 byte long.
__mxs__calc_adler32	Calculates the adler32 and returns the result. Returned as an 8 byte long. If the return value is -1 or -2 then the value is being calculated, check back again for the calculated value a few seconds later.
__mxs__calc_md5	Calculates the MD5 of the object and returns the result. Returned as a byte array. If the return value is an empty array then the value is being calculated, check back again for the calculated value a few seconds later.
__mxs__vaultquotabytes*	Returns the maximum capacity allowed of the vault. Returned as an 8 byte long.
__mxs__vaultspaceusedbytes*	Returns the current space used by the vault. Returned as an 8 byte long.
__mxs__userHasTag=*	Returns if the current user has been tagged with the given key. As of server version 2.4: “dropspot admin” is supported as a tag.
__mxs__userCapabilities*	Returns the current user’s access capabilities. Returns a string composed of “R” “W” “D” “S” and “U” as appropriate.
__mxs__online	Since v3. It returns if the object is online, e.g. it has not been stubbed and the data is accessible.

* When searching on an attribute to return one of these values, use object ID “00000000-0000-0000-0000-000000000001-1”

To retrieve one of the listed values perform a get metadata attribute API command on the appropriate object with the listed key.