IBM Cloud Object Storage System™
Version 3.14.0

Accesser Application Guide

IBM

This edition applies to IBM Cloud Object Storage System™ and is valid until replaced by new editions.

Contents

Chapter 1. Configure prerequisites for the Accesser Application

The following items are prerequisites for the Accesser Application.

The IBM Cloud Object Storage Accesser® Application package includes this documentation and installers. It runs locally to provide access to a Vault via HTTP interface. A traditional HTTP client or Simple Object Connector can be used to communicate with the Accesser® Application over SOH, CSO, or OSOS APIs. The Simple Object Connector is provided separately including its Java API documentation. The Simple Object Connector is a library that is API-compatible with the DSAF SDK but instead communicates with a system via a configured Accesser® Application Application.

Note: This feature is not compatible with Concentrated Dispersal (CD).

Meet physical installation requirements

The following items are physical installation requirements for the Accesser application.

Table 1. Memory requirements

Component	Setup	Minimum	Recommended
Memory	IBM Cloud Object Storage Vault™ Only	4.0 GB	16.0 GB
Memory	Using Vault Mirrors	8.0 GB	16.0 GB

Table 2. Other requirements

Disk space	60 GB minimum recommended
CPU	2 cores

Configure memory settings

Configure memory settings to use the Accesser® Application.

Three Accesser® Application features impact the amount of memory required for a JVM to run the library successfully:

Table 3. Features

Feature	Description	Requirement
Optimistic Write	Smooths out performance issues slow cause. It buffers Slices to be sent to those slow Slicestor® Nodes.	Increase memory allotment to improve Optimistic Write performance.
Liveliness Monitoring	Ensures that clients do not wait for a connection that has timed out due to network issues or server failures. It disconnects from a Slicestor® Node if it has not read any data over a given amount of time.	Run the JVM without pauses longer than the disconnect threshold; it is set currently to 30 seconds.

Table 3. Features (continued)

Feature	Description	Requirement
Concurrency	Additional clients of the Accesser® Application require added memory for connection tracking, more in-transit data, and more session management.	Increase memory allotment to improve concurrency.

Java's Garbage Collector occasionally needs to perform "stop-the-world" GCs. If the amount of memory allocated to the library is too large when this occurs, the GC will take longer than the disconnect threshold. While the larger amount of memory helps the optimistic write feature, it will cause the "liveliness monitoring" feature to erroneously disconnect all Slicestor® Nodes. This will cause in-progress I/O operations to fail, though future operations will succeed.

The Accesser® Application allocates memory in two ways:

- Allocating objects to the JVM heap.

  Java uses the Heap to store most objects and scans it for orphans during a full GC.
- Allocating binary data directly to semi-managed space.

The maximum for both types of memory allocation can be set separately.

IBM Cloud Object Storage System™ recommends the memory settings in the following table as a good balance for two common configurations. For both configurations, both heap size and direct memory size are set. These configurations are command line arguments passed to the JVM before indicating application arguments.

Further tuning is occasionally required for some system configurations.

Table 4. Recommended memory configurations

Deployment Scenario	System Memory (GB)	JVM Heap (GB)
Server Class Systems	16+	4
Client Class Systems	4+	1
Server Class Systems (Using Vault Mirrors)	32+	8
Client Class Systems (Using Vault Mirrors)	8+	2

These settings meet the requirements for most use cases.

Note: For scenarios where the file size is large (> 10 GB) and/or the number of concurrent uploads/downloads is large (> 50), contact Customer Support for additional guidance.

Use a certified software configuration

The Accesser® Application must use one of each of the listed operating systems, JVMs and native code packages to run properly. Native code packages are optional, but they do improve the performance of the Accesser® Application

Table 5. Certified Software Components compatible with the Accesser® Device Application

Component	Certified Versions
Operating System	• Red Hat Enterprise Linux 6.3 and 6.4 • Debian 7

Table 5. Certified Software Components compatible with the Accesser® Device Application (continued)

Component	**Certified Versions**
Java™ Virtual Machines	Oracle Java SE 7 or later or OpenJDK RE 7 or later
Native Code Packages (Optional)	• Linux x86-64 Operating System • OpenSSL dynamic library `libssl.so.0.9.8` (part of `openssl-0.9.8`) or `libssl.so.1.0.1` (part of `openssl-1.0.1`)

Chapter 2. Installing the Accesser Application

Select an appropriate version of the Accesser Application, choose an installation method, and run the install package to install the Accesser Application.

Select an appropriate version of the Accesser Application

Accesser® Application versions are targeted for use with a specific release.

When reading data from a system, clients applications must use a version equal to or later than the one used to write data. Later versions may use codecs unavailable in earlier versions.

CAUTION:
Except where otherwise noted, upgrading the Accesser® Application should resolve any read issues.

Select an Accesser Application installation method

Except where otherwise noted, upgrading the Accesser® Application can be installed using one of the installers provided with this package.

Table 6. Accesser® Application bundles

Package	Build Filenames
Connectors Library	`dsnet-connector-3.7.1.zip`
Accesser® Application Archive Bundle	`dsnet-accesser-3.7.1.zip` `dsnet-accesser-3.7.1.tar.gz`
Accesser® Application Debian Installer	`dsnet-accesser-3.7.1.deb`
Accesser® Application Red Hat Package Installer	`dsnet-accesser-3.7.1.rpm`

Run the install package

Install the Accesser® Application according to the appropriate OS and package.

Table 7. Varieties of UNIX packages to install Accesser® Application

Target OS	Installation Command
Debian-based Linux	`sudo dpkg dsnet-accesser-3.7.1.deb`
Red Hat Enterprise Linux	`sudo rpm dsnet-accesser-3.7.1.rpm`
Debian-based Linux	`tar -xvf dsnet-accesser-3.7.1.tar.gz`

Review where files were installed

Review the application archive, package installer, and simple object connector library destinations.

Application archive destinations

For installations that unpacked the archive file, the following directories are included in the bundle.

Table 8. Directories contained in the Accesser® Application Archive Bundle

Path	Purpose
`bin/`	Launch script and binaries
`conf/`	Configuration files
`lib/`	jar binaries
`log/`	Default log location
`pki/`	Cryptographic material when PKI authentication is configured
`run/`	information about active Accesser® Application or Applications

Package installer destinations

When using a system-specific package installer, the files and directories integrate into the destination OS and are found in the locations specified in the following table.

Table 9. Destination Directories for the Accesser® Application Red Hat and Debian Package Installers

Purpose	Red Hat Path	Debian Path
Binaries	`/opt/dsnet-accesser/bin/`	`/usr/bin/`
Launch script	`/etc/init.d/dsnet-accesser/`	`/etc/init.d/dsnet-accesser/`
Configuration files	`/etc/dsnet-accesser/`	`/etc/dsnet-accesser/`
jar binaries	`/opt/dsnet-accesser/lib/`	`/usr/lib/dsnet-accesser/lib/`
Default log location	`/var/log/dsnet-accesser/`	`/var/log/dsnet-accesser/`
Default PKI location	`/opt/dsnet-accesser/pki/`	`/var/lib/dsnet-accesser/pki/`
Information about active Accesser® Applications	`/var/run/dsnet-accesser/`	`/var/run/dsnet-accesser/`

Simple object connector library destinations

There is documentation contained in the binary Simple Object Connector library build.

Table 10. Documentation contained in the binary Simple Object Connector library build

File or Directory	Description
`Javadoc`	Java API documentation for Simple Object Connector
`lib/`	Simple Object Connector Binaries

Configure the network

The Accesser® Application connects to Manager-provided APIs, the registry (provided by any Slicestor® Node) and all Slicestor® Nodes.

The Registry: The registry contains records of network devices, user accounts, Vault bootstrapping data and vault-to-Slicestor® Node mappings.

Table 11. Port usage for regular Accesser Node use

Destination	Port	Purpose
Slicestor® Nodes	`TCP 5000`	Data operations and registry lookup

Table 11. Port usage for regular Accesser Node use (continued)

Destination	Port	Purpose
Slicestor® Nodes	`TCP 7 (OPEN or REJECT)`	Round trip time calculation
Manager Node	`TCP 443`	Vault usage query via HTTPS Manager API

Note: TCP port 7 may be closed, but set any firewall rules to send `REJECT` messages and not drop packets.

Chapter 3. Configuring the Accesser Application

To configure the Accesser Application, edit the configuration file, change the memory settings, configure authentication, change the port number an default API type, change the vault list, and support multiple instances.

Edit the configuration file

Editing the `dsnet-accesser.conf` file allows the operator to set options within the Accesser® Application.

Unless specified, all property changes take effect dynamically without requiring a restart of the Accesser® Application.

The configuration file is located at:

- `/etc/dsnet-accesser` for Debian and Red Hat installations
- `conf` subdirectory for unpacked tape archives (tar file)

Run the startup script with the `-f` flag to use a different set of configuration files:

```
dsnet-accesser #–f# {path}
```

Accesser® Application requires the configuration parameters in the table below to function properly.

See *Accesser® Application Conf File Settings* for additional details.

Table 12. Minimum configuration parameters for Accesser® Application

Parameter	Description	Example
`vault`	Comma-separated list of Vault names to access via Accesser® Application	`[vault1,vault2]`
`login`	Username for user who can access the specified vault. Optional if Vault has an anonymous access enabled or if PKI authentication enabled. When using anonymous, this parameter should be commented out.	`{username}`
`password`	Password for specified username.	`{password}`
`bootstrap_locations`	A list of URIs containing the hostname or IP and port of Slicestor® appliances on which the specified Vault is deployed CAUTION: `bootstrap_location` (no *s*) has been deprecated. Use this instead.	`["dsnet://{ipOrHostName1}:{port1}", "dsnet://{ipOrHostName2}:{port2}"]`
`manager_uri`	HTTPS URI containing the hostname or IP of the manager appliance	`(https://{ipOrHostName})`
`virtual-host-suffix`	Virtual host suffix used for virtual host style bucket addressing syntax	`example.com`

Change memory settings

Heap size and direct memory size can be set on systems with varied memory requirements.

To accommodate different system sizes, memory is allocated as a percentage of total system memory by default. The following properties override this behavior. Both `memory` and `off_heap_memory` are limited to less than 1,000 MB.

Attention: Restart of the Accesser® Application after changing memory settings.

Table 13. List of settings in the `[launcher]` section of the Accesser Application configuration file (dsnet-accesserconf)

Setting	Description	Metric	Common Value
`memory`	Set the heap size to a fixed value. An equal amount of memory is allocated for off-heap memory if not configured otherwise.	MB	4000 or 16000
`off_heap_memory`	Amount of off-heap memory to use. Both on- and off-heap memory are required for proper operation and performance.	MB	1000 or 4000
`memory_percent`	Percent of system memory to use when configuring memory dynamically. **Note:** An equivalent amount will be used for off-heap memory unless configured otherwise.	Decimal equivalent of percent	0.25 (25%)
`off_heap_memory_percent`	Percent of system memory to use when configuring off-heap memory dynamically. Both on-heap and off-heap memory are required for proper operation and performance.	Decimal equivalent of percent	0.25 (25%)
`max_memory`	Maximum memory the daemon should allocate for its heap when configuring memory dynamically. **Note:** An equivalent amount will be used for off-heap memory unless configured otherwise. It is recommended this value not exceed 32000 as larger heaps disable Java compressed OOPs optimization which may impact performance.	MB	32000
`max_off_heap_memory`	Maximum memory the daemon should allow to be allocated for off-heap memory allocations when configuring memory dynamically. Both on-heap and off-heap memory are required for proper operation and performance.	MB	32000

Note: See *Change Memory Settings* for the configuration file listing.

Configure authentication

Configure authentication for the Accesser Application.

Configure PKI and bootstrap authentication

To configure the Accesser® Application PKI, certain settings must be configured.

A user who has an X.509 certificate and matching private key can perform PKI authentication if the certificate is assigned to a user account on the storage network. The Manager Web Interface can be used to assign PKI identities to user accounts.

The Accesser® Application can be configured with a host certificate, private key, and the certificate or certificates of CAs that the Accesser® Application should consider trustworthy.

When the Accesser® Application is provided with the bootstrap URI, it will connect to the given Slicestor® Node and download the configuration information. Unless Accesser® Application has been configured with trusted certificates, the default behavior is to not authenticate the remote server's identity.

To enable bootstrap URI server authentication and ensure the configuration information's authenticity, a trusted CA certificate is required. The path to the file or directory containing one or more trusted certificates should be configured in the `dsnet-accesser.conf` file.

The trusted certificate verifies the certificate of the Slicestor® Node during the TLS handshake which occurs during the configuration information request.

CAUTION: Restart the Accesser® Application after changing the certificate settings.

Table 14. List of settings in the `[device]` section of the Accesser Application configuration file (`dsnet-accesser.conf`)

Setting	Description	Required	Value
`host_certificate`	The PEM format X.509 certificate for this device.	Yes. If not specified, this node cannot accept SSL/TLS connections or perform certificate-based authentication.	`./pki/certs/localhost.crt`
`host_key`	The host key for this device in PEM format.	Yes, if a host certificate is provided.	`./pki/private/localhost.key`
`trusted_certs`	A directory or file containing trusted device or CA certificates in PEM format.	Yes, if verification of storage network configuration will occur.	`./pki/certs/ca-dsnet.crt`
`trusted_crls`	A directory or file containing CRLs from trusted CAs in PEM format.	Yes or CA revocations cannot be detected.	`./pki/crl/ca-dsnet.crl`
`trusted_certs`	A directory or file containing trusted device or CA certificates in PEM format.	Yes, if a host certificate is specified.	`/var/lib/dsnet-accesser/pki/certs/`

Note: See *PKI and Bootstrap Authentication* for the configuration file listing.

Change port number and default API type

The following properties define the address and port or ports to which Accesser® Application binds, as well as the API exposed on the root endpoint.

Table 15. List of settings in the [service] section of the Accesser® Application configuration file (`dsnet-accesser.conf`)

Setting	Description	Value
`http.bind_port`	Ports to bind the HTTP gateway to. This should be in the form of [{`ports`}], where {`ports`} is a comma-separated list of integers in the valid port range.	`[8080]`
`http.bind_address`	Hostname or IP address to which the application should be bound. • For a local-only server, use `localhost`. • To listen on all interfaces, use `0.0.0.0`.	`localhost`
`http.default_protocol`	The API type to expose on the root endpoint of the http service. Acceptable values include s3 (default), soh, and openstack.	`s3`

Note: See *Accesser® Application Conf File Settings* for the configuration file listing.

Change vault list

The Vault list is one property that defines the list of Vault names that the Accesser® Application will expose.

Table 16. List of settings in the `[access]` section of the Accesser® Application configuration file (`dsnet-accesser.conf`)

Setting	Description	Value
`vault`	List of Vault names this Accesser® Application should serve up	[{vault1}, {vault2}]

Note: See ***Vault List*** for the configuration file listing.

CAUTION: When deleting a Vault from the system, remove the corresponding entry from the list of vaults. If a new Vault is created with the same name, remove the Vault name entry from the configuration first then add it again, or the Accesser® Application process must be restarted, before the new Vault can be used.

Support multiple instances

Each instance of the Accesser® Application after the first needs its own configuration file with non-conflicting HTTP ports.

About this task

Specify the location of this configuration file using the `-f` flag.

Procedure

1. Start with the default configuration file:

```
/etc/init.d/dsnet-accesser start
```

2. Start a second Accesser® Application with a second configuration file:

```
/etc/init.d/dsnet-accessser -f /etc/dsnet-accesser/{dsnet-accesser.second.conf} start
```

Chapter 4. Using Accesser Application

The Accesser® Application can be used with traditional HTTP clients or Simple Object Connector.

It runs on localhost, port `8080` by default.

Note: See *Change Port Number and Default API Type* on how to change the port number.

Start the Accesser Application

Start the Accesser Application from a `deb` and `rpm` install, or from a tape archive install.

Start from a deb and rpm install

Start the Accesser® Application:

```
/etc/init.d/dsnet-accesser start
```

Start from a tape archive install

Procedure

1. Change to the directory into which Accesser® Application has been extracted.
   ```
   cd {directory}
   ```
2. Start the Accesser® Application.
   ```
   ./bin/dsnet-accesser start
   ```

API compatibility

All APIs supported by the Accesser® Device are also supported by the Accesser® Application.

Table 17. Accesser® Application Supported API Versions

API	Supported Version	Further information
Simple Object over HTTP	2.5	*Simple Object HTTP API 2.5 Developer Guide*
OpenStack (Swift)	1.0	*OpenStack Object Storage API 1.0 Developer Guide*
Cloud Storage Object (S3-Compatible)	2.4	*Cloud Storage Object API 2.4 Developer Guide*

Note: Accesser® Nodes do not support all features of the compatible APIs. For a full list of the supported features, see each API guide.

Use simple object vaults

This package provides fully documented interfaces for the Simple Object Vault.

Application developers may create embedded clients using the Simple Object Connector API or may use a traditional HTTP client to access deployed vaults. These vaults can be accessed through a server running SOH via an Accesser® Application.

Note: See the *Simple Object HTTP API 2.5 Developer Guide* for more information.

Use a traditional client

The Accesser® Application can accept the same methods as other types of Accesser® Nodes.

Note: See the *API Guides* for further details.

Use the simple object connector

If the Simple Object Connector targets a properly-configured Accesser® Application, the standard Simple Object HTTP operations can be performed without further Vault or credential configuration.

Note: See *Working through the Application Life Cycle* for a code sample for creating the Simple Object Connector.

Chapter 5. Upgrade a Java-based application from DSAF

Applications written against the DSAF API must be changed to use the Simple Object Connector library.

The Simple Object Connector requires significantly less client code to perform operations against a Simple Object Vault compared to using the DSAF client.

Once an application has been updated to use the Simple Object Connector, there should be no further need to rebuild an application to support updates to the Simple Object Connector except to support additional APIs as they become available.

Note: See *Working through the Application Life Cycle* for a code sample necessary for setting up a client to perform operations against a configured Accesser® Application.

Chapter 6. Working through the application life cycle

A client application using the Accesser® Application will perform data operations on a particular type of vault.

Once an Accesser® Application is configured properly (see *Configuring the Accesser Application*), clients can begin communicating with the Accesser® Application using the SOHConnector or by one of the supported APIs over HTTP. With the `dsnet-connector jar` on the application's `classpath`, the `SOHConnector` can be created for writing objects to the Accesser® Application using its configured Vault credentials:

```
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Collections;
import org.cleversafe.connectors.soh.SOHConnector;
final SOHConnector connector = new SOHConnector(Collections.singletonList(new URL(
"http://localhost:8080/soh/vault")));
```

With a configured Accesser® Application and a constructed `SOHConnector` on the same device, all SOH Vault operations (e.g. read, write, list, delete) perform with no further need to configure credentials or load vaults.

Note: See the `SOHConnector` javadoc, installed on the client with the package, for more detailed developer information.

Set quotas

The Manager Node can configure Vaults with both a soft and hard quota.

When a soft quota is exceeded, the Manager Node triggers an alert, but the Accesser® Application does not.

When a hard quota is exceeded, the Manager Node tells the client to stop writing data. If the Manager Node is unavailable, a client cannot enforce a hard quota; it continues to write data even if the hard quota will be exceeded.

The Accesser® Application stops writing new objects when the hard quota is exceeded, but any outstanding writes continue. The Accesser® Application attempts to track the size of writes it accepts to avoid exceeding the hard quota even if it cannot receive usage updates from the Manager Node. Due to the distributed nature of a system, it is not always possible to precisely enforce hard quotas without the Manager Node if multiple clients are performing writes simultaneously.

Slicestor® Nodes do not enforce a hard quota as write requests are received. If a hard quota has been exceeded significantly, the Manager Node removes all write permission grants from the Vault to lock it. The Slicestor® Nodes prevents further writes to the affected Vault. In exceptional cases, this "hard lockout" condition may require administrator intervention to resolve.

Query a vault with the `sohconnector`

Usage statistics can be queried on a per-Vault basis using the `VaultUsageMonitor` interface.

The `SOHConnector` interface extends the `SimpleObjectController` interface.

The `SimpleObjectController` interface extends `VaultUsageMonitor` interface.

This interface estimates logical source data size. Each number excludes any IDA-added overhead. Internally, actual usage statistics are aggregated from information that each Slicestor® Node tracks individually. The aggregated figures may fluctuate if rebuild activity is ongoing. Actual data collection is performed at system determined poll periods. Both data aggregation and polling period delays may lead to inaccurate estimates.

Two sets of numbers are available:

- Numbers that reflect possible allocation if no hard quota were set
- Numbers that show actual possible allocation taking into account a hard quota

If no hard quota is set, these numbers are the same. In most cases, API methods should use numbers that reflect any hard quota.

Note: If an Accesser® Application displays total share size to an end user, it should use the quota-based numbers to give the user a reasonable expectation of when there will be no more space available.

Chapter 7. Limitations

There are some limitations to monitoring the Accesser application and additional items to consider

User authentication ignored when using APIs

All communication between the Accesser® Applicationand the system is authenticated using the credentials specified in the Accesser® Application configuration file, using either username/password, PKI, or anonymous authentication.

When accessing Accesser® Application using one of the available HTTP REST APIs, any user authentication provided in the request will be ignored.

Client-side access control bypassed

The Accesser® Application runs in a mode where any client-side authorization checks are bypassed.

Access Control enforcement is performed by the Slicestor® Devices based on the Vault permissions granted to the user specified in the Accesser® Application configuration file.

Object ACLs not checked or enforced

The Accesser® Application does not check or enforce Object ACLs.

The permissions for the Vault where the object resides determines access to that Object. Object ACLs cannot grant permissions beyond what the Vault permission allows when accessed through the Accesser® Application.

Vault ACLs not checked after vault is loaded

The Vault ACL for a Vault is loaded when the Accesser® Application is first started or when a new Vault is added to the configuration file.

If the Vault permissions of the user configured for Accesser® Application are revoked, the existence of the Vault and its ACL contents are still visible to the Accesser® Application until the application is restarted.

Cross-origin resource sharing only available to users with full vault permissions

On an Accesser® Application, Cross-Origin Resource Sharing (CORS) functionality can be configured only for users with `FULL_CONTROL` permission on the given Vault.

As the Accesser® Application cannot enforce client-side authorization, I/O operations can be executed with these caveats:

- `GET` Bucket CORS operations are allowed if the user has `READ` permissions on the Vault.
- `PUT` and `DELETE` Bucket CORS operations are allowed if the user has `WRITE` permissions on the Vault.

No objects ownership recorded

No object owner is recorded in the metadata for objects written via the Accesser® Application.

Unsupported operations in CSO

Some CSO API operations are unsupported when performed through the Accesser® Application.

Note: For a list of supported operations, see ***Operations on Buckets*** in the *Cloud Storage Object API Guide*.

Property loading

Changes to some properties within the Accesser® Application are picked up dynamically, while changes to others require restarting the Accesser® Application.

When changing the following properties, restart the Accesser® Application process:

- Username/Password
- PKI Credentials (Host Certificate, Host Key, Trusted CA, CRL)
- Statistics bind port

Chapter 8. Error scenarios

Since the Accesser® Application provides an HTTP interface, errors are returned to traditional HTTP clients as HTTP status codes.

Accesser® Application logs can be examined after an HTTP error code is returned to the client to further diagnose why a particular status code was returned to a client.

For long running operations, a failure can occur immediately when the request is made or when the operation is in progress. In some cases, an exception will be thrown from a different object. This depends on the state of the operation.

Note: On a read request, a Java Exception will be thrown from the `read()` method if the initial request fails. If the request succeeds initially, an InputStream is returned. This does not guarantee that further errors do not occur. If an error does occur a Java Exception is thrown from the read method of the input stream.

Some write methods take an input stream that must be implemented by the calling code and contain the source data to be written. If a failure occurs due to an I/O error in the source data input stream, the `write()` method will throw a distinct exception to indicate the source of the problem clearly. During normal operation, all SDK methods will only throw checked exceptions.

I/O request error scenarios

Table 18. Possible scenarios for I/O request errors

Scenario	Read Request	Write Request	Delete Request
Vault does not exist	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Invalid Vault name	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Vault is deleted	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Object does not exist	No Data Object Exception	Success	False Returned
Write exceeds available space	Not applicable	Storage I/O Exception	Not applicable
Write exceeds allowed hard quota	Not applicable	Object Quota Exception	Not applicable
Less than threshold numbers of slices or Slicestor® Nodes available	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Source data input stream fails	Not applicable	I/O Exception	Not applicable
Subject does not have permission	Security Exception	Security Exception	Security Exception

In-flight request error scenarios

Table 19. Possible Scenarios for In Flight Errors

Scenario	Read Request	Write Request	Delete Request
Vault does not exist	Initial request fails	Initial request fails	Initial request fails

Table 19. Possible Scenarios for In Flight Errors (continued)

Scenario	Read Request	Write Request	Delete Request
Invalid Vault name	Initial request fails	Initial request fails	Initial request fails
Vault is deleted	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Object does not exist	Initial request fails	Success	Initial request fails
Write exceeds available space	Not applicable	Storage I/O Exception	Not applicable
Write exceeds allowed hard quota	Not applicable	Success (unchecked during writes)	Not applicable
Less than threshold numbers of slices or Slicestor Nodes available	Storage I/O Exception	Storage I/O Exception	Storage I/O Exception
Source data input stream fails	Not applicable	I/O Exception	Not applicable
Subject does not have permission	Initial request fails	Initial request fails	Initial request fails

Chapter 9. Troubleshooting

You can troubleshoot the Accesser Application.

Logging

The Accesser® Application comes with extensive logging capabilities.

Logs can be configured via the `log4j2.xml` file.

Note: For details about the format of this file, see *Apache Log4j 2 Configuration*.

Table 20. Accesser® Application log files

Log File	Contents	Format
`stdout.log`	Log messages printed to standard error and standard output streams of the Accesser® Application. This will only contain log messages related to launch and termination of the Accesser® Application or error messages that prevent Accesser® Application from launching.	
`platform.log`	Log messages generated by the Accesser® Application during normal runtime. This file only contains events of log levels `INFO`, `WARN` and `ERROR`. The default configuration for Accesser® Application sets the default log level to `INFO` to suppress more verbose log messages.	
`debug.log`	Log messages generated by the Accesser® Application during normal runtime. This file can contain events of any log level but only if the logging level is at least `DEBUG` level severity. The default configuration for Accesser® Application sets the log level to `INFO`; logging to this file is suppressed. If the `dsnet-accesser.conf` file is modified to set `[logging] level=debug`, more verbose log messages are written to this log file which may be useful for troubleshooting issues with the Accesser® Application.	
`http.log`	HTTP log messages for requests serviced by the Accesser® Application.	Apache httpd log
`access.log`	Access log messages containing more detailed information about requests serviced by the Accesser® Application.	JSON

Table 20. Accesser® Application log files (continued)

Log File	Contents	Format
`report.log`	Periodic statistical information about various aspects of the Accesser® Application.	JSON

Note: By default, all log files roll over based on a configured maximum net size for each log file.

Chapter 10. Accesser Application `conf` file settings

You can configure settings in the dsnet-accesser.conf file.

Memory settings

Memory settings are set in launcher section of the dsnet-accesser.conf file.

```
[launcher]
# To accomodate variable system sizes, memory is by default
# allocated as a percentage of total system memory. The
# following properties override this behavior. Both memory
# and off_heap_memory are limited to less than 1,000 MB

# Statically set the heap size to a fixed value. An equal
# amount of memory is also allocated for off-heap memory if
# not configured otherwise.
#memory={memory in mb, usually 4000 or 16000}

# Amount of off-heap memory to use. Both on-heap and off-heap
# memory are required for proper operation and performance.
#off_heap_memory={memory in mb, usually 1000 or 4000}

# percent of system memory to use when configuring memory
# dynamically. Note that an equivalent amount will be
# used for off-heap memory unless configured otherwise above.
memory_percent=0.25

# percent of system memory to use when configuring off-heap
# memory dynamically. Both on-heap and off-heap memory
# are required for proper operation and performance.
off_heap_memory_percent=0.25

# max memory that the daemon should allocate for its heap
# when configuring memory dynamically. Note that an
# equivalent amount will be used for off-heap memory unless
# configured otherwise. It is recommended for this value to
# not exceed 32000 as larger heaps will disable Java's
# compressed OOPs optimization which may impact performance.
max_memory=32000

# max memory that the daemon should allow to be allocated
# for off-heap memory allocations when configuring memory
# dynamically. Both on-heap and off-heap memory are required
# for proper operation and performance.
max_off_heap_memory=32000
```

PKI and bootstrap authentication

PKI and bootstrap authentication are set in the device section of the dsnet-accesser.conf file.

```
[device]

# The PEM format X.509 certificate for this device. If not specified
# this device will be unable to accept SSL/TLS connections or perform
# certificate-based authentication.
#host_certificate = ./pki/certs/localhost.crt

# The host key for this device in PEM format. Required if host
# certificate is provided.
#host_key = ./pki/private/localhost.key
```

```
# A directory or file containing trusted device or CA certificates in PEM
# format. Required if host certificate is specified.
#trusted_certs = ./pki/certs/ca-dsnet.crt

# A directory or file containing CRLs from trusted CAs in PEM format.
# If not specified, CA revocations will not be detected.
#trusted_crls = ./pki/crl/ca-dsnet.crl

# A directory or file containing trusted device or CA certificates in PEM
# format. Required if host certificate is specified.
trusted_certs = /var/lib/dsnet-accesser/pki/certs/
```

Port number and API type

Port number and API type are set in the service section of the dsnet-accesser.conf file.

```
[service]

# Ports to bind the HTTP gateway to. This should be in the form of [<ports>],
# where <ports> is a comma-separated list of integers in the valid port range.
http.bind_port=[8080]

# Hostname or IP address to bind to. For a local-only server, use localhost.
# To listen on all interfaces, use 0.0.0.0
http.bind_address = localhost

# Default API type.
# Acceptable values include s3, soh, and openstack.
# http.default_protocol = s3
```

Vault list

Vaults can be listed in the access section of the dsnet-accesser.conf file.

```
[access]

# list of Vault names this accesser application should serve up
vault=[vault1,vault2]
```

S3 virtual host suffix

The virtual host suffix can be set in the s3 section of the dsnet-accesser.conf file.

```
[s3]
# virtual host suffix used for virtual host style bucket addressing syntax
virtual-host-suffix = example.com
```

Chapter 11. Glossary

Glossary of terms used in this document.

Abbreviation	**Term**
ACL	Access Control List
API	Application Programming Interface
CA	Certificate Authority
codec	coder-decoder
CRL	Certificate Revocation List
CSO	Cloud Storage Object
DSAF	Dispersed Storage Access Framework
dsNet	Dispersed Storage Network
GC	garbage collection
HTTP	HyperText Transmission Protocol
I/O	Input and Output
IP	Internet Protocol
jar	Java Archive
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
OS	Operating System
OSOS	OpenStack (Swift) Object Storage
PEM	Privacy Enhanced Mail
PKI	Public Key Infrastructure
REST	Representational State Transfer
HTTPS	Secure HyperText Transmission Protocol
SSL	Secure Sockets Library
SOC	Simple Object Connector
SOH	Simple Object over HTTP
SDK	Software Development Kit
tar	Tape Archive
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier

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
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

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

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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 the names and addresses used by an actual business enterprise 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

Printed in USA