

IBM StoredIQ Connector  
API SDK Version 1.0.1

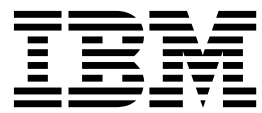
## *Reference Guide*





IBM StoredIQ Connector  
API SDK Version 1.0.1

*Reference Guide*



**Note**

Before using this information and the product it supports, read the information in Notices.

This edition applies to Version 7.6.0.14 of product number 5724M86 and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2017, 2018.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

## Table of contents

<b>About this publication</b> . . . . .	<b>v</b>	Connection management . . . . .	8
IBM StoredIQ Platform product library . . . . .	v	Data object traversal . . . . .	9
Contacting IBM StoredIQ customer support . . . . .	v	Attribute management. . . . .	10
Contacting IBM . . . . .	v	Content Management . . . . .	12
How to send your comments . . . . .	vi		
 <b>Chapter 1. Connector API SDK overview</b> . . . . .	<b>1</b>	 <b>Chapter 5. Sample NFS Connector.</b> . . . .	<b>17</b>
 <b>Chapter 2. Version check enforcement</b> . . . . .	<b>3</b>	Analysis of data source . . . . .	17
 <b>Chapter 3. Setting up and integrating</b>		API implementation . . . . .	17
<b>Connector API SDK.</b> . . . . .	<b>5</b>	 <b>Notices</b> . . . . .	<b>21</b>
 <b>Chapter 4. Connector API SDK</b> . . . . .	<b>7</b>	Privacy policy considerations . . . . .	23
StoredIQ code shared with the user. . . . .	7	Terms and conditions for product documentation. . . . .	24
Connector API. . . . .	7	Trademarks . . . . .	24
		 <b>Index</b> . . . . .	<b>27</b>



---

## About this publication

*IBM® StoredIQ Connector API SDK Reference Guide* provides information about the IBM StoredIQ Connector API SDK, its installation, utility, and various administrative and data scripts.

---

## IBM StoredIQ Platform product library

The following documents are available in the IBM StoredIQ Platform product library.

- *IBM StoredIQ Platform Deployment and Configuration Guide*, SC27-6386
- *IBM StoredIQ Platform Overview Guide*, GC27-6398
- *IBM StoredIQ Platform Data Server Administration Guide*, SC27-5692
- *IBM StoredIQ Administrator Administration Guide*, SC27-5688
- *IBM StoredIQ Data Workbench User Guide*, SC27-5691
- *IBM StoredIQ eDiscovery User Guide*, SC27-5693
- *IBM StoredIQ Policy Manager User Guide*, SC27-5694

---

## Contacting IBM StoredIQ customer support

For IBM StoredIQ technical support or to learn about available service options, contact IBM StoredIQ customer support at this phone number:

### Support and assistance

- 1-866-227-2068

To e-mail IBM StoredIQ customer support, use this email address:

- [storediqsupport@us.ibm.com](mailto:storediqsupport@us.ibm.com)

For information about how to contact IBM, see the Contact IBM web site at <http://www.ibm.com/contact/us/>

### IBM Knowledge Center

The IBM StoredIQ publications can be found from IBM Knowledge Center.

### PDF publications

The IBM Publication Center site offers customized search functions to help you find all the IBM publications you need.

## Contacting IBM

For general inquiries, call 800-IBM-4YOU (800-426-4968). To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

For more information about how to contact IBM, including TTY service, see the Contact IBM website at <http://www.ibm.com/contact/us/>.

---

## How to send your comments

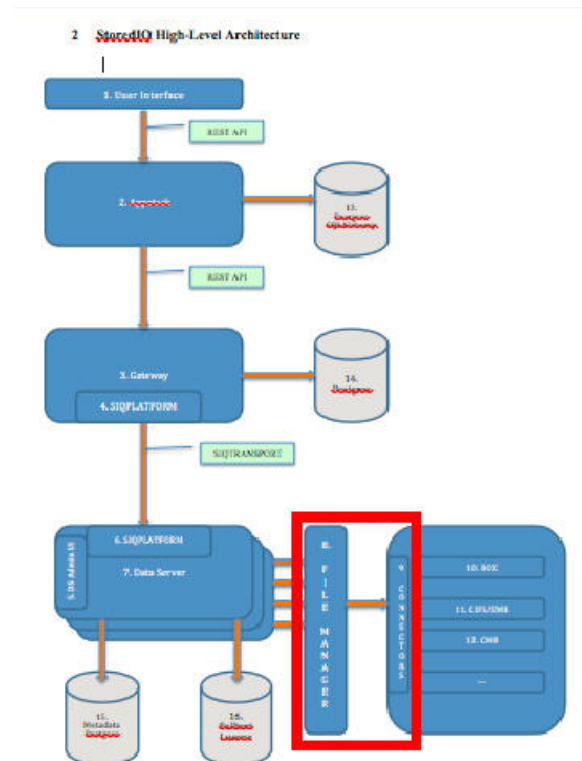
Your feedback is important in helping to provide the most accurate and highest quality information.

Send your comments by using the online reader comment form at [https://www14.software.ibm.com/webapp/iwm/web/signup.do?lang=en\\_US&source=swg-rcf](https://www14.software.ibm.com/webapp/iwm/web/signup.do?lang=en_US&source=swg-rcf).



## Chapter 1. Connector API SDK overview

A Connector is a software component of IBM StoredIQ that is used to connect to a data source such as a network file system and access its data. Using IBM StoredIQ Connector API SDK that is represented by the red box in the following diagram, developers of other companies can develop Connectors to new data sources outside IBM StoredIQ development environment. These Connectors, then, can be integrated with a live IBM StoredIQ application to index, search, manage, and analyze data on the data source.



The Connector API is designed to support data sources ranging from traditional file systems such as NFS to modern ones, such as IBM Connections. Modern data sources vary significantly from traditional data sources in terms of data object properties and organization of data objects within the data source. For example, in a NFS file system, data objects have a limited set of attributes and are organized in a tree structure. However, in modern file systems, like those hosting social networking content, attributes of data objects are non-trivial and the relationship and traversal are specific to the data source.

To handle the wide range of data source types, all operations that are run by the IBM StoredIQ application on data objects are condensed into a small set of primitive APIs. These APIs are divided into four groups:

- Connection management
- Data object traversal management
- Attribute management
- Content access management

These APIs are described in detail in “Connector API” on page 7. All that a Connector developer is required to do is to implement these APIs. These APIs include `listDir()` for traversal, `lstat()` for managing data object properties, validating data object path, and so on. Attribute management support takes care of managing additional properties or attributes of data objects in a modern data source.

Two significant advantages of the IBM StoredIQ Connector API SDK are as follows.

1. It simplifies Connector development by cleanly decoupling Connector logic from the IBM StoredIQ application logic, therefore, speeding up Connector development to new data sources.
2. It prevents Connector specific logic from creeping into Data Server logic, which causes additional testing overhead and unforeseen issues.

The Connector API SDK also includes code that implements a fully working NFSn based sample Connector. Chapter 3 describes the sample Connector and provides guidelines about the development of a Connector to a new data source.

**Note:**

- This version of SDK supports only Python language.
- Users of this SDK are assumed to have thorough knowledge of IBM StoredIQ operation.
- The functional description of the API may change in future versions.

---

## Chapter 2. Version check enforcement

IBM StoredIQ Connector API SDK Version 1.0.1 is the first release version that implements version compatibility check with IBM StoredIQ.

The version check feature can result in the following behavior.

- The registration script, `/usr/local/storediq/bin/register_connector.py`, which is run on the Gateway or data server matches the version in `<user_connector>/sdk_version.py` with the installed IBM StoredIQ.
- If the version numbers mismatch, the stack trace displays an error message:  
Exception: `[Reg_type.INVALID_SDK_VERSION]` SDK Version check failed!
- If the version files mismatch, the stack trace displays an error message:  
Exception: `[Reg_type.INVALID_VER_MODULE]` [Errno 2]. No such file or directory.. Reasons for the failure can be integrating IBM StoredIQ Connector API SDK Version 0.1.0 compatible user-Connector with IBM StoredIQ version 7.6.0.14 or later.



---

## Chapter 3. Setting up and integrating Connector API SDK

To work with the Connector API SDK, follow these steps.

### About this task

- Information in this section is subject to change in future releases.
- Python version  $\geq 2.6$  and setup tools are expected to be installed on the development system.
- SDK administration involves manual steps for this release.
- This version of Connector API SDK is compatible with IBM StoredIQ Release version 7.6.0.14 and all subsequent releases and modifications until otherwise instructed in the new editions.

### Procedure

1. Download IBM StoredIQ Connector API SDK software from the IBM website.  
Download the appropriate SDK archive, `connector_sdk-1.0.1.tar.gz` or `connector_sdk-1.0.1.zip` from the IBM website into any folder on the development system and extract its contents. For example, this folder can be `/home/<user-name>/dev` on your Linux system, or `/Users/<user-name>/dev` on your OSX, or `C:\Users\<user-name>\dev` on your Windows system.

2. Install the StoredIQ Connector API SDK software on the development system.

On Linux and OSX, enter these commands:

- `cd <dev-folder-path>`
- `tar xvzf connector_sdk-1.0.1.tar.gz`
- `cd connector_sdk-1.0.1`
- `python setup.py install --install-lib='my_install' --connector-dir='my_connector'`

#### Note:

- The SDK is installed in `'my_install'`.
- `'my_install/my_connector'` is the name of your connector folder.
- If you omit the `--install-lib` option, then `my_connector` is installed in the current directory.
- If you omit the `--connector-dir` option, then the Connector folder is named as `user_connector`.

3. Develop the IBM StoredIQ Connector by using Connector API SDK.
  - See information in next two chapters.
  - Your Connector code must be placed in the `my_install/my_connector` folder.
  - Avoid changes to `'my_install/my_connector/sdk_version.py'`.
  - This module is provided for version compatibility enforcement.
  - Avoid changes in the shared code in the `siq_connector` folder.
4. Integrate the IBM StoredIQ Connector with live IBM StoredIQ application.

**Note:** For the current release, this step involves a simple copy action.

Copy the folder that contains your Connector code to each Data Server (DS) and Gateway (GW). For example, the command to execute on your

development system can be `scp -rp <my_connector> root@<IP-address-of-DS-or-GW>/usr/lib/python2.6/site-packages`

**Note:** From 7.6.0.13 onward, every DS contains a `siq_connector` folder to start with. Any additional libraries that are used by your Connector must be located in the `<my_connector>/external` folder.

5. Register the new Connector with the live IBM StoredIQ application.
  - On each data server and gateway,
    - a. `cd` to `site-packages`.
    - b. Run the command `python32 /usr/loca/storediq/bin/register_connector.py -c <class-path> [-w 'yes' | 'no']`.  
For example, if your Connector folder is `my_connector`, class name is `MyConnector` and the module that defines the class is `my_module`, then class-path is `'my_connector.my_module.MyConnector'` (quotes included).  
For the `-w` option, if it is specified, it indicates that the Connector is a read-write Connector.  
Run the command `python32 /usr/loca/storediq/bin/register_connector.py -h` for more information.
    - c. Perform this step only if your Connector uses Custom attributes. See `sample_attributes.py` about defining custom attributes.  
Assume that the name of the custom attributes that is defined in `my_attributes.py`, which is similar to `sample_attributes.py`, is `MyConnObj`. Verify that the registration is successful by checking the database:
      - `psql -U dfuser dfdata`
      - `Select appliance_attribute_id, attribute_name from application_schema.attributes where attribute_name like '%MyConnObj%';`
      - `\q`You can see in the output that the attribute `SampleConnObjFileType` (see `sample_attributes.py`) is as follows:

appliance_attribute_id	attribute_name
2091001	SampleConnObjFileType (nfs-template)
    - d. Restart services on the data server and gateway by using this command:  
`service deepfiler restart`

---

## Chapter 4. Connector API SDK

The following topics describe each of the Connector API and code that are common to StoredIQ core application and the Connector.

---

### StoredIQ code shared with the user

Connector API SDK shares the following Python modules with the user. These modules mainly contain the default implementation and the utility functions for some of the methods.

From StoredIQ Release 7.6.0.13 on, these shared modules come preinstalled in the `/usr/lib/python2.6/site-packages/siq_connector` folder.

**Note:** Users must not change these modules.  
Description of each module contents is as follows.

#### **templateconnection.py**

This module contains the API definitions along with default implementations, where necessary. See more details in the following section.

#### **node.py**

This module contains Node class definition, which manages data objects and their properties.

During indexing the objects in the connector, or harvesting as it is called in StoredIQ, the object traversal methods talk to each other. The importance of node is explained in the following data object traversal section. This module is also self-documented. See the embedded comments for each method to find out how the methods need to be overridden if needed.

#### **templateexceptions.py**

This module contains exception class definitions that are common to StoredIQ. You can use exceptions in this module while you implementing `templateconnection` or `node` methods.

#### **attributedefs.py**

This module contains definitions to construct custom attributes. The standard set of attributes (properties) of a data object that is identified by StoredIQ includes size, access times, and more.

For more information, see comments in `attributedefs.py`. For information on how to construct a custom attribute of a data object, see `sample_attributes.py`.

#### **sdk\_version.py**

This module is used for version compatibility enforcement.

---

## Connector API

The Connector API is divided into four groups as described in the following sections. The user-defined Connector class derives from the `TemplateConnection` class that defines the Connector Interface. The user must provide the implementation for methods for which a default implementation is not provided.

## Connection management

The APIs in this group manage connections to the data source. Developers can use the appropriate network protocol, such as HTTPS, CIFS, NFS mount, to implement the `connect()`, `disconnect()`, and `shutdown()` APIs.

**`__init__(self, serverName, userName, userPwd, options, serial, mountPoint)`**

This constructor method provides information that is collected by using the Add Volume UI dialogue.

**serverName**

[string] contains the IP address or hostname of the server that houses the data store.

**userName**

[string] contains the username that is used in authentication.

**userPwd**

[string] contains the password that is used in authentication.

**options**

[string] contains connector-specific extra options. In a sample connector, the NFS share is passed as 'share=<share-string>'. For example, share=/deepfs/demo-A. Multiple options must be separated by a semi-colon (;).

**serial** Unused.

**mountPoint**

[string] Mount path as constructed by StoredIQ, which is used for NFS like data sources. It is the default if mount=<mount-path> is absent in options string.

**Return**

N/A

**Default Implementation**

N/A

**`connect(self)`**

Use the information that is provided by the constructor method to establish a connection with the server that hosts the data source.

**Returns**

True if connection is successfully established, otherwise raise an appropriate `TemplateException`.

**Default Implementation**

Not provided.

**`disconnect(self)`**

Disconnect from the server that hosts the data source.

**Returns**

True if disconnection is successful, otherwise `False`.

**Default Implementation**

Not provided.

**`shutdown(self)`**

Releases the Connector instance and the associated StoredIQ resources. The default implementation calls `disconnect()`. This method needs to be implemented only if `disconnect()` is not sufficient to perform cleanup.



**Returns**

True if shutdown is successful, otherwise False.

**Default Implementation**

Provided.

## Data object traversal

APIs in this group implement data source traversal. By default, StoredIQ traverses data source tree in pre-order (depth first) pattern. `list_dir()` is the most important API of the group that facilitates the complete traversal by presenting data objects that consist both directory and non-directory types on each invocation.

This ability of APIs, `list_dir()` and `lstat()` that when it is combined with the built-in walker component, suffices to traverse most data sources that range from traditional to most modern data sources. It is for this reason that users are not provided the option to implement their own walker class.

**get\_list\_dir\_page\_size(self)**

Returns the page size that is used in calls to `list_dir()`.

**Returns:**

Maximum number of entries that the `list_dir()` function can return.

**Default Implementation**

Provided.

**list\_dir(self, node)**

This method lists the files and directories in the specified directory. Each element in the list must include a node object to the parent element specified by the node.

**node** [Node] points to the directory object whose children are to be listed.

**Returns:**

List of nodes.

**Default Implementation**

Not provided.

**list\_dir\_next(self, node)**

This method retrieves the next `itemCount` items from the list of items that is provided by most recent `list_dir()` call.

**node:** [Node] points to the directory object whose children are to be retrieved.

**Returns:**

List of nodes.

**Default Implementation**

Not provided.

**path\_exists(self, path)**

This method determines whether the given path exists. The default implementation uses `lstat()` to determine the existence. If it is sufficient, this method does not need to be overridden.

**path:** [string] points to the data object whose existence is to be determined.

**Returns:**

True if the object pointed to by path exists, otherwise it returns False.

**Default Implementation**

Provided.

## Attribute management

APIs in this group manage custom attributes.

By default, StoredIQ collects the standard attributes or properties of a data object that include size, access times, and more. However, attributes of data objects in a modern data source are not trivial. APIs in this section provide a mechanism for the user to define custom attributes and register them with the StoredIQ application.

For more information about creating custom attributes, see comments in `sample_attributes.py` and `attributedefs.py`.

**get\_attribute\_def(cls)**

This class method returns a dictionary that consists definitions of custom attributes. The default returns an empty dictionary that indicates no custom attributes for data objects.

**Returns**

[dictionary] contains custom attribute definitions that pertain to data objects. An empty dictionary indicates the absence of custom attributes.

**Default Implementation**

Provided.

**get\_audit\_attr\_names(self, path)**

Return a list of attribute names to be included in the audit. When audit is needed, StoredIQ looks for these names in the `lstat` extras. If the name is found, the value is included in the audit information. If the value is not found in extras, then a blank value is shown. If no attributes audit besides the standard file system attributes, this method returns an empty list. The default implementation is to return an empty list.

**path:** [string]

**Returns:**

List of attribute names or empty list.

**Default Implementation**

Provided.

**get\_fs\_name(cls)**

The name that is returned by this class method shows up in the add volume dropdown in the UI. It must match the `fs_name` in `attribute_def`.

**Returns:**

[string] Connector name.

**Default Implementation**

Not provided.

**get\_node(self, path, extras)**

Returns an object of Node class with given parameters. Node class provides the behavior and functions for a single object for a connector. The

default node class that is imported is Node. However, a connector developer is free to override this behavior. In most cases, each connector has a specific behavior for its node. The default Node implements behavior for a generic Linux like connector.

**path:** [string] Points to data object to be initialized in Node.

**extras:** [dict] Object properties that are saved in Node.

**Returns:**

[Node] Node object that is instantiated with object properties.

**Default Implementation**

Provided.

**is\_dir(self, path="")**

Method to check whether a given path is an existing directory. It is implemented by calling `lstat()` and check the stat to see whether the path is a directory. If any exception is raised by `lstat()`, it is assumed that the path is not a valid directory.

**path:** [string] Points to data object that is to be checked for a directory.

**Returns:**

True if it is a directory and False otherwise.

**Default Implementation**

Provided.

**is\_read\_only(self, path)**

Checks if a file at the given path is read-only. If the implementation cannot tell for the path, it returns False if the system is not read only. It is used in move and delete action.

**path:** [string] points to data object whose read and write access is to be determined.

**Returns:**

True if the object pointed to by path is read-only, otherwise it returns False.

**Default Implementation**

Not provided.

**lstat(self, path)**

This method retrieves the file system-specific attributes for the specified file. This method emulates the `os.lstat()` of the Python library. The returned value is of type `posix.stat_result`. This object might be accessed as a tuple of mode, ino, dev, nlink, uid, gid, size, atime, mtime, ctime or through the attributes `st_mode`, `st_ino`, `st_dev`, `st_nlink`, `st_uid`, and more.

**path:** [string] points to the data object whose properties are to be retrieved.

**Returns:**

[tuple] containing `posix.stat_result`, which contains file system properties for the data object.

**Default Implementation**

Not provided.

**lstat\_extras(self, path, extras)**

Get the file system and the extra attributes for the specified file.

**path:** [string] points to the data object whose properties are to be retrieved.

**Returns:**

[tuple] containing `posix.stat_result` and `Node`.  
`posix.stat_result` contains file system properties for the data object.

**Default Implementation**

Not provided.

## Content Management

APIs in this group manage the content that is retrieved from the data objects.

**close\_file(self, filehandle)**

Closes the file that is represented by the file handle. `filehandle` can be an object of a type, `Handle`, refer to `sample_conn.py`, which contains Connector-specific handle information, such as OS file handle. This object is opaque to the caller. For files that are open in write mode, if they are available at the time of close, it returns the stat and stat extras. If these files are not available, they are retrieved from the repository as part of finishing write operations. These files are used to fill the audit and attribute information. If the stats are not available, an empty dictionary must be returned. In read mode files, the stats are not used.

**filehandle:**

[Handle] File handle that is created by using `open_file` or `create_file`.

**Returns:**

Dictionary empty or one containing stat and stat extras.

**Default Implementation**

Not provided.

**create\_file(self, path, size, attrMap)**

Creates a file that is specified by the path on the data source if the data source is not read-only. If the file exists, `IOError` is raised. The attributes to be associated with the file are passed in. The same attributes are also passed into `close_file` for convenience since some connectors require the attributes on Close and others on Create.

**path:** [string] Specifies the fully qualified path to the data object to be created on the data source.

**size** [integer] Unused.

**attrMap**

[dict] Attributes that are associated with the file object.

**Returns:**

[Handle] A file handle object. It is opaque and implementation dependent. It must be passed to calls to `write_file` and `close_file`.

**Default Implementation**

Not provided.

**get\_unsupported\_characters(self)**

Specify a list of characters that are not supported by the Connector. This list is used on a copy to action to escape characters in source file paths that this destination Connector does not support. By default this list is empty.

**Returns:**

[list] A list of unsupported characters on filenames.

**Default Implementation**

Provided.

**makedirs(self, path)**

Creates a directory to include one for every segment that is missing in the path.

**path:** [string] Fully qualified path of a directory to be created.

**Returns:**

True if it is successful; otherwise, it returns False. If an IO error occurs, it raises an exception.

**Default Implementation**

Not provided.

**open\_file(self, path, mode=FILE\_READ)**

Opens a file that is specified by a path on the data source. If the file cannot be opened, IOError is raised. Supports only modes, FILE\_READ, and FILE\_WRITE. An existing file that is open for FILE\_WRITE truncates the file. If the mode is omitted, it defaults to FILE\_READ. If the mode is FILE\_READ and the file does not exist, IOError is raised. If an error occurs when the file is opened, an IOError is raised.

**path:** [string] Specifies the fully qualified path to the data object to be opened on the data source.

**mode:** [constant] File open modes that are defined in TemplateConnection class.

**Returns:**

[File handle] A file handle object. It is opaque and implementation dependent. It must be passed to calls to write\_file and close\_file.

**Default Implementation**

Not provided.

**pre\_process(self, path, fileHandle)**

Gets path and file handle of the read file, manipulates the file and returns the handle if the file needs to be manipulated before rendering. This file handle is the IBM StoredIQ file handle and it is not the same as the file handle class that is previously described.

**path:** [string] Points to the data object whose contents are to be processed before rendering.

**fileHandle:**

[SIQ Handle] Points to SIQ handle, a special object that is provided by the caller. The user must return this filehandle as is after pre\_process() is complete.

**Returns:**

[SIQ Handle] A file handle that is provided by the caller to be returned as is.

**Default Implementation**

Provided.

**read\_file(self, path, filehandle, readSize)**

Reads from a file. Read <readSize> bytes from the file (less if the read hits

EOF before obtaining size bytes). Reads from the current position of the file handle. If readSize is > 0, this number of bytes is read, less if it hits EOF. If readSize is = 0, then a 0 length buffer is returned. If readSize is < 0, the whole file is read from the offset to the end. If the fileHandle is at or past the EOF, a 0 length buffer is returned. If an error occurs when reading from the file, an IOError is raised.

**path:** [string] Specifies the fully qualified path to the data object to be read from.

**filehandle:**

[Handle] File handle that is created by using open\_file or create\_file.

**readSize**

[integer] Byte count to be read.

**Returns:**

Number of bytes that is read.

**Default Implementation**

Not provided.

**rmdir(self, path)**

Removes a directory that is specified by the path.

**path:** [string] Fully qualified path of directory to be removed.

**Returns:**

True if it is successful; otherwise it returns False. If an IOError occurs, an exception is raised.

**Default Implementation**

Not provided.

**unlink(self, path)**

Deletes a file that is specified by the path.

**path:** [string] Fully qualified path of a file to be deleted.

**Returns:**

True if it is successful; otherwise it returns False. If an IO Error occurs, an exception is raised.

**Default Implementation**

Not provided.

**validate\_directories(self, path, startDir, endDir)**

Validates that the initial directory, the start directory, and the end directory are valid for a volume definition. If not specified in the configuration, these directories can be set to None. It is up to the file manager implementation to determine whether None values are valid. Returns a list of error messages that are found when the directories are checked. Returns None or an empty collection if all of the directories are valid. If any exception is raised by the implementation, it is assumed that the directories are not valid.

**path:** [string] Relative path of Initial Directory as specified in the Add Volume UI.

**startDir:**

[string] Relative path of Start Directory as specified in the Add Volume UI.

**endDir:**

[string] Relative path of End Directory as specified in the Add Volume UI.

**Returns:**

A list that contains the collection of error messages; otherwise it is an empty list.

**Default Implementation**

Not provided.

**validate\_directories\_in\_file\_mgr(self)**

Behavior flag to say whether the file manager validates its own directories that are specified in the volume definition. If this method is overridden to return True, then the `validate_directories` method must be implemented.

**Returns:**

True if users select to validate on their own; otherwise, it returns False.

**Default Implementation**

Provided.

**verify\_mount(self, path="", include\_dirs=None)**

This method takes in a path that is associated with the volume definition and validates that this path is readable and writable. It also returns a list of any errors that are encountered. By default, this implementation calls `chkpath(path)` to determine whether a path is readable and calls `is_read_only` to determine whether the path is writable. It is up to the Connector implementation to decide how to most efficiently determine it. Many implementations use calls to `lstat()` and `list_dir`. By default, the verified flag is true unless an exception checks readability. Failures in writability are treated as warnings, thus the verification flag is set to true if the write test fails.

**path:** [string] Relative path that needs to be validated.

**includeDirs:**

[string] Unused.

**Returns:**

[Tuple] in the form of `verified, errlist, readable, writeable`, where `verified` is a Boolean flag to indicate verification failure. If False, it is assumed that the mount is not valid and an error is shown by `StoredIQ`. `errlist` is a list that contains error messages. `readable` is boolean that indicates readability. If False, it causes add or update volume to fail. `writeable` is boolean that indicates writeability.

**Default Implementation**

Provided.

**verify\_mount\_in\_file\_mgr(self)**

A behavior flag to indicate whether the file manager validates its own mount by returning whether the mount is readable, writable, or whether errors access the mount. If this method is overridden to return True, then the `verify_mount` method must be implemented.

**Return:**

True if users select to verify on their own; otherwise, it returns False.

**Default Implementation:**

Provided.

**write\_file(self, path, filehandle, filebuf)**

Writes the data present in the input filebuf buffer by using the file handle (filehandle) that is passed in.

**path:** [string] Specifies the fully qualified path to the data object to be written to.

**filehandle:**

[opaque filehandle] A file handle that is created by using open\_file or create\_file.

**filebuf:**

[string] A file buffer that contains data to be written.

**Returns:**

Number of bytes that is written.

**Default Implementation:**

Not provided.



---

## Chapter 5. Sample NFS Connector

The following topics discuss methodology that is involved in the development of a sample Connector to the NFS v3 file system.

---

### Analysis of data source

Through analysis of the NFS data source, the four groups of APIs can be best implemented as follows.

#### Connection management

Use the mount and umount commands to manage connections to the file system. The required information, such as mount point, share name, server name, and more, are passed by using the Add Volume dialog. For more information, see the implementation of connect() method for Sample Connector in “API implementation.”

#### Data object traversal

The file system that is used for support through NFS has the tree structure. Hence, use the tree traversal that is provided by the default walker class. The list\_dir() method is implemented so that it populates objects whose paths are relative to the sub-tree path that is provided. The sample connector also supports the initial\_directory feature, when specified in the Initial Directory box of the Add Volume dialog.

#### Attribute management

The data objects of NFS file system do not possess any special attributes that are different from the file system metadata collected by StoredIQ by default. However, to make use of the custom attribute support by the SDK, a custom attribute SampleConnObjFileType is incorporated. The value of this attribute is computed in the form of mime type that represents the type of data object contents.

For more information, see \_get\_extras() method, in “API implementation.” For details about how to create custom attribute, see comments in sample\_attributes.py and attributedefs.py..

#### Content management

Fully utilize the operating system support to retrieve content from the data objects. No additional pre or post processing is needed on the content.

---

### API implementation

Implementation details of some important APIs are explained as follows.

#### connect() [Connection Management]

Use the Add Volume dialog to collect the information that is required for connection management. If the share is already mounted, unmount it first and then remount it. In the Add Volume dialog, **Source Type** contains the name of the connector nfs-template. Enter **localhost** for Server and share=/deepfs/demo-SK in the **Option String** box to identify the share name. A mount-point, if any, can be specified in the **Option String** text box, which now appears as share=/deepfs/demo-SK;mount=/usr/my\_mount. If it is specified, the remote volume is connected to or mounted on the path that is specified by mount option. Otherwise, the default path is used.

Note that the **Class Name** field is set to `sample_connector.sample_conn.SampleConnector`, where `sample_connector` is the folder in `/usr/lib/python2.6/site-packages` on the data server. `sample_conn` is the name of the module in the `sample_connector` folder and contains the `SampleConnector` class, which implements the sample NFS connector.

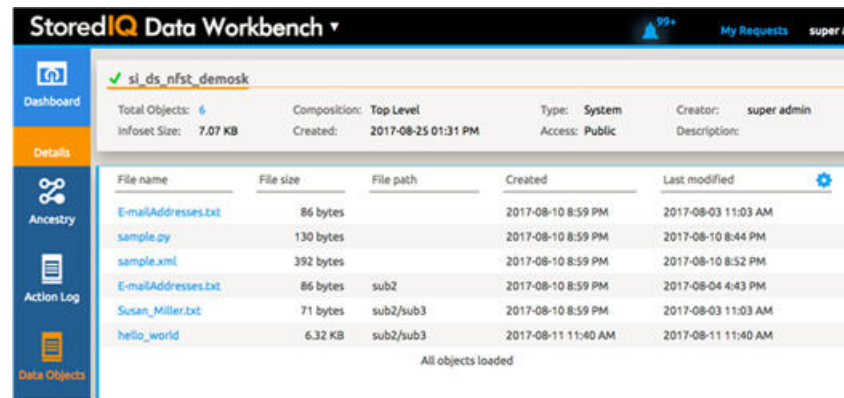
On the test NFS volume, data objects are organized as follows. **sub1** in the **Initial Directory** indicates that data objects are processed inside `sub1` folder and under its level.

```
./CompanyName.txt
./hello_world.c
./sub1
./sub1/E-mailAddresses.txt
./sub1/sub2
./sub1/sub2/sub3
./sub1/sub2/sub3/hello_world
./sub1/sub2/sub3/Susan_Miller.txt
./sub1/sub2/E-mailAddresses.txt
./sub1/sample.xml
./sub1/sample.py
```

### list\_dir() [Data Object Traversal]

The walker component of StoredIQ traverses the data source by calling `list_dir()`, which returns the data objects along with relative paths to the walker. For example, when `list_dir()` is called with `path='sub2'`, then `list_dir()` returns the data objects present at sub2 level, which include sub3, a folder, and E-mailAddresses.txt.

The data objects **Details** page on StoredIQ Data Workbench presents the object name and paths correctly as shown in the following image.



File name	File size	File path	Created	Last modified
E-mailAddresses.txt	86 bytes		2017-08-10 8:59 PM	2017-08-03 11:03 AM
sample.py	130 bytes		2017-08-10 8:59 PM	2017-08-10 8:44 PM
sample.xml	392 bytes		2017-08-10 8:59 PM	2017-08-10 8:52 PM
E-mailAddresses.txt	86 bytes	sub2	2017-08-10 8:59 PM	2017-08-04 4:43 PM
Susan_Miller.txt	71 bytes	sub2/sub3	2017-08-10 8:59 PM	2017-08-03 11:03 AM
hello_world	6.32 KB	sub2/sub3	2017-08-11 11:40 AM	2017-08-11 11:40 AM

### \_get\_extras() [Attribute Management]

The `_get_extras()` local method that is called by `list_dir()` builds the mime-type attribute that is representative of the content. The Attribute Summary report looks as follows:

	A	B	C	D	E	F	G	H
1	Attribute Name	Count	Attribute P Description					
2	SampleConnObjFileType (nfs-template)/	6	/Library/At Type of sample Connector object type					
3	Language	5	/Library/At The primary language of the content of an object.					
4	Content error code	2	/Library/At When the content processing state is "Partial" or "Failed,"					
5	Binary error code	1	/Library/At When binary processing has been performed, the followi					
6								



---

## Notices

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

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

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

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

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.

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

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

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

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

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 actual people or business enterprises is entirely coincidental.

#### COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: © Copyright IBM Corp. 2004, 2015. All rights reserved.

---

## Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

[Depending upon the configurations deployed, this Software Offering may use [session] [and] [persistent] cookies that collect each user's [name,] [user name,] [password,] [profile name,] [position,] or [other personally identifiable information] for purposes of [session management,] [authentication,] [enhanced user usability,] [single sign-on configuration] [or other usage tracking or functional purposes.] These cookies [cannot be disabled] [can be disabled, but disabling them will also [likely] eliminate the functionality they enable].

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

---

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

---

## Trademarks

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.



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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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



---

# Index

## A

- administering Connector API SDK 5
- analysis of data source 17
- API 1
- API definition 7
- API implementation 17
- attribute management 10, 17
- Attribute Summary report 17
- audit 10
- audit attribute 12

## C

- connection management 8, 17
- Connector 7
- Connector API 7
- Connector API SDK 1
- Connector code 5
- Connector instance 8
- Connector interface 8
- content management 12, 17
- custom attribute 10, 17
- custom attributes 7

## D

- data object traversal 17
- data source 17
- data source traversal 9

## E

- exception class definition 7

## F

- file handle 12

## L

- legal
  - notices 21
  - trademarks 24

## M

- mime 17
- mode 12

## N

- NFS v3 17
- Node class definition 7
- notices
  - legal 21

## S

- sample NFS connector 17
- sample NFS Connector 17
- shared modules 7

## T

- trademarks 24
- tree traversal 17

## U

- user-defined Connector 8

## W

- walker class 9







Printed in USA