# Add File

The **Add File** operation creates a new file metadata record under the specified namespace.

**Resource Type:** File, Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/add

**Request Parameters:**

The following parameters apply to both *file* and *directory* type.

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| type | Required. Type of the record. Either 0 for file or 1 for directory. |
| owner | Required. ID of the user who created the file. |
| sequence | Required. Unique value that identifies this operation. If subsequent request with the same value is received when the current operation has already completed, the request will be discarded. |
| path | Required. Full path of the file/directory including the file/directory name. |

The following parameters apply to only *file* type.

|  |  |
| --- | --- |
| Name | Description |
| size | Required. Length of the file in bytes. If it is a newly created file without data, the value 0 can be used. If the value is more than 0, the request body must content the information on the file slices. |
| blksize | Required. Length of the portion of data within the file for which the checksum is calculated in bytes. If verifying the integrity of the data is not required, the value given will be 0; else the value given must be a common multiple of the quorum used in the IDA (information dispersal algorithm) and 512 (minimum page size in a Windows Azure page blob). |
| key | Optional. A secure random string used in the calculation of the checksum. |
| version | Required. The IDA (information dispersal algorithm) version that is used to split and combine the file data. Corresponds to an actual matrix. |

**Request Body:**

The request body applies to only *file* type. The content is made up of name value pairs with each pair on a new line. If the file is newly created without content (the value of size request header is 0), then the request body is empty.

|  |  |
| --- | --- |
| Name | Description |
| sliceCount | Required. Number of slices for this file. |
| sliceSeq<n> | Required. Sequence of the slice within all slices. <n> denotes the index of the slice. |
| sliceName<n> | Required. File name of the slice. <n> denotes the index of the slice. |
| sliceServerId<n> | Required. ID of the file slice store where the content of the slice is actually stored. <n> denotes the index of the slice. |
| sliceLength<n> | Required. Length of the content in the slice. <n> denotes the index of the slice. |
| sliceChecksum<n> | Optional. The aggregated checksum computed on the content of the slice. <n> denotes the index of the slice. |
| sliceSegCount<n> | Required. Number of segments within the slice. Segments exist because a portion of the data fails to write to the slice store and thus have to be stored at another slice store temporarily. <n> denotes the index of the slice. |
| sliceSeg<n> sliceName<i> | Optional. The segment name. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceServerId<i> | Optional. ID of the file slice store where the content of the segment is actually stored. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceSegOffset<i> | Optional. Starting position of the content in the segment within the slice. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceSegLength<i> | Optional. Length of the content in the segment. <n> denotes the index of the slice and <i> denotes the index of the segment. |

**Sample Request:**

|  |
| --- |
| Directory Request Syntax:  http://localhost:9010/file/add?namespace=urn:nyp.edu.sg&type=1&owner=victoria&sequence=12345678910&path=%2Fabc  File Request Syntax:  http://localhost:9010/file/add?namespace=urn:nyp.edu.sg&type=1&owner=victoria&sequence=12345678910&path=%2Fabc%2sample.txt&size=100&blksize=102400&key=sd2sd5&version=1  File Request Body:  sliceCount=2  sliceSeq0=1  sliceName0=slice1  sliceServerId0=tfs1  sliceLength0=20  sliceChecksum0=dfg445j645kjsfkyre230jkdfgcioe  sliceSegCount0=1  sliceSeg0sliceName0=seg1  sliceSeg0sliceServerId0=tfs2  sliceSeg0sliceSegOffset0=20  sliceSeg0sliceSegLength0=40  sliceSeq1=2  sliceName1=slice2  sliceServerId1=tfs2  sliceLength1=60  sliceChecksum1=j8374jsd873sdi2mim823f932ssdq  sliceSegCount1=0 |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If the name for the file or directory already exists, status code 409 (Conflict) is returned.

If the parent path of the file or directory is not found, status code 400 (Bad request) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The date and time the metadata record is created in milliseconds since January 1, 1970, 00:00:00 GMT.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  1282817514297 |

# Update File

The **Update File** operation updates an existing file metadata record under the specified namespace.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/update

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| by | Required. ID of the user who updated the file. |
| sequence | Required. Unique value that identifies this operation. If subsequent request with the same value is received when the current operation has already completed, the request will be discarded. |
| filename | Required. Full path of the file including the file name. |
| size | Required. Length of the file in bytes. |
| blksize | Required. Length of the portion of data within the file for which the checksum is calculated in bytes. If verifying the integrity of the data is not required, the value given will be 0; else the value given must be a common multiple of the quorum used in the IDA (information dispersal algorithm) and 512 (minimum page size in a Windows Azure page blob). |
| key | Optional. A secure random string used in the calculation of the checksum. |

**Request Body:**

The content is made up of name value pairs with each pair on a new line.

|  |  |
| --- | --- |
| Name | Description |
| sliceCount | Required. Number of slices for this file. |
| sliceSeq<n> | Required. Sequence of the slice within all slices. <n> denotes the index of the slice. |
| sliceName<n> | Required. File name of the slice. <n> denotes the index of the slice. |
| sliceServerId<n> | Required. ID of the file slice store where the content of the slice is actually stored. <n> denotes the index of the slice. |
| sliceLength<n> | Required. Length of the content in the slice. <n> denotes the index of the slice. |
| sliceChecksum<n> | Optional. The aggregated checksum computed on the content of the slice. <n> denotes the index of the slice. |
| sliceSegCount<n> | Required. Number of segments within the slice. Segments exist because a portion of the data fails to write to the slice store and thus have to be stored at another slice store temporarily. <n> denotes the index of the slice. |
| sliceSeg<n> sliceName<i> | Optional. The segment name. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceServerId<i> | Optional. ID of the file slice store where the content of the segment is actually stored. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceSegOffset<i> | Optional. Starting position of the content in the segment within the slice. <n> denotes the index of the slice and <i> denotes the index of the segment. |
| sliceSeg<n> sliceSegLength<i> | Optional. Length of the content in the segment. <n> denotes the index of the slice and <i> denotes the index of the segment. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/update?namespace=urn:nyp.edu.sg&by=victoria&sequence=12345678910&filename=%2Fabc%2sample.txt&size=150&blksize=102400&key=sd2sd5  File Request Body:  sliceCount=2  sliceSeq0=1  sliceName0=slice1  sliceServerId0=tfs1  sliceLength0=20  sliceChecksum0=dfg445j645kjsfkyre230jkdfgcioe  sliceSegCount0=1  sliceSeg0sliceName0=seg1  sliceSeg0sliceServerId0=tfs2  sliceSeg0sliceSegOffset0=20  sliceSeg0sliceSegLength0=40  sliceSeq1=2  sliceName1=slice2  sliceServerId1=tfs2  sliceLength1=60  sliceChecksum1=j8374jsd873sdi2mim823f932ssdq  sliceSegCount1=0 |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If the file is currently open for write related operations by another user, status code 423 (Locked) is returned.

If the parent path of the file or directory is not found, status code 400 (Bad request) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The date and time the file metadata record is updated in milliseconds since January 1, 1970, 00:00:00 GMT.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  1282817514297 |

# Move/Rename File

The **Move File** operation changes the logical path and/or name of an existing file metadata record under the specified namespace.

**Resource Type:** File, Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/move

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| old\_namespace | Required. Namespace the file currently belong to. |
| sequence | Required. Unique value that identifies this operation. If subsequent request with the same value is received when the current operation has already completed, the request will be discarded. |
| old\_filename | Required. Full path of the existing file including the file/directory name. |
| namespace | Required. Namespace the file will be moved to. |
| path | Required. Full path, including the changed file/directory name the file will be moved to. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/move?namespace=urn:nyp.edu.sg&old\_filename=%2Fabc%2sample.txt&namespace=urn:nyp.edu.sg&path=%2Fsample.txt&sequence=12345678910 |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If the directory to be renamed is the root directory, or the path the file/directory to move to already exist, status code 409 (Conflict) is returned.

If the current namespace or the file/directory is not found, or the parent path of the file/directory to move to does not exist, status code 400 (Bad request) is returned.

If the file is currently open for write related operations by another user, status code 423 (Locked) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The date and time the file metadata record is last accessed in milliseconds since January 1, 1970, 00:00:00 GMT.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  1282817514297 |

# Delete File

The **Delete File** operation removes an existing file metadata record under the specified namespace.

**Resource Type:** File, Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/delete

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| type | Required. Type of the record. Either 0 for file or 1 for directory. |
| path | Required. Full path, including the file/directory name of the file to be deleted. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/delete?namespace=urn:nyp.edu.sg&path=%2Fsample.txt&type=0 |

Response

The response includes an HTTP status.

**Status Code:**

A successful operation returns status code 200 (OK).

If the path provided does not correspond to the type, status code 400 (Bad request) is returned.

If the file is currently open for write related operations by another user, status code 423 (Locked) is returned.

For more information on other status codes, refer to Appendix.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK |

# Lock File

The **Lock File** operation locks an existing file metadata record under the specified namespace. It prevents another user from doing write related operations like move, delete etc.

If the user locking the file does not refresh the lock by invoking the same request within a defined interval, the lock will expire.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/lock

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| by | Required. ID of the user attempting to obtain the lock. |
| filename | Required. Full path, including the name of the file to lock. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/lock?namespace=urn:nyp.edu.sg&filename=%2Fsample.txt&by=victoria |

Response

The response includes an HTTP status.

**Status Code:**

A successful operation returns status code 200 (OK).

If the path of the file provided does not exist, status code 400 (Bad request) is returned.

If the path provided is not a file, status code 409 (Conflict) is returned.

If the file is currently open for write related operations by another user, status code 423 (Locked) is returned.

For more information on other status codes, refer to Appendix.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK |

# Unlock File

The **Unlock File** operation releases an previously lock obtained on a file.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/unlock

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| by | Required. ID of the user attempting to release the lock. |
| filename | Required. Full path, including the name of the file to unlock. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/unlock?namespace=urn:nyp.edu.sg&filename=%2Fsample.txt&by=victoria |

Response

The response includes an HTTP status.

**Status Code:**

A successful operation returns status code 200 (OK).

If the path of the file provided does not exist, status code 400 (Bad request) is returned.

If the path provided is not a file, status code 409 (Conflict) is returned.

If the file is not currently locked by the user provided, status code 423 (Locked) is returned.

For more information on other status codes, refer to Appendix.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK |

# Get File

The **Get File** operation retrieves an existing file metadata record under the specified namespace.

This function can be used to check if a file/directory exists. If no such record exists, then the name value pairs returned in the response body will be mostly blank.

**Resource Type:** File, Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/get

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| path | Required. Full path, including the name of the file/directory to retrieve. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/unlock?namespace=urn:nyp.edu.sg&path=%2Fsample.txt |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If the namespace provided does not exist, status code 400 (Bad request) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The content is made up of name value pairs with each pair on a new line. The following values are returned even if the metadata record does not exist.

|  |  |
| --- | --- |
| Name | Description |
| idaVersion | If the record exists, then the value is the IDA (Information dispersal algorithm) version used (for file type only); else, it would be the current IDA version used. |
| idaMatrix | The matrix of the specified IDA version in the form of “|” and “,” delimited values. |
| idaShares | The number of slices generated by the specified IDA version. |
| idaQuorum | The minimum number of slices of the specified IDA version required to get the original file data. |

The following values are returned if the metadata record exists.

|  |  |
| --- | --- |
| Name | Description |
| type | Type of the metadata record. Either 0 for file or 1 for directory. |
| owner | ID of the user who created the file/directory. |
| creation | Creation date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT. |
| lastModified | Last updated date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT. |
| lastAccessed | Last accessed date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT. |

The following values are returned if the metadata record belongs to a *file* type.

|  |  |
| --- | --- |
| Name | Description |
| size | The length of the file in bytes. |
| blkSize | Length of the portion of data within the file for which the checksum is calculated in bytes. If verifying the integrity of the data is not required, then this name value pair will not be returned. |
| keyHash | A secure random string used in the calculation of the checksum. If verifying the integrity of the data is not required, then this name value pair will not be returned. |
| fileMode | Flag to indicate if the file is in the mist of mode change. -1 to denote no mode change; 1 to denote changing from non-streaming to streaming; 0 to denote changing from streaming to non-streaming. |
| sliceCount | Number of slices for this file. |
| sliceSeq<n> | Sequence of the slice within all slices. <n> denotes the index of the slice. |
| sliceName<n> | File name of the slice. <n> denotes the index of the slice. |
| sliceServerId<n> | ID of the file slice store where the content of the slice is actually stored. <n> denotes the index of the slice. |
| sliceLength<n> | Length of the content in the slice. <n> denotes the index of the slice. |
| sliceChecksum<n> | The aggregated checksum computed on the content of the slice. <n> denotes the index of the slice. This name value pair only exists if verifying the integrity of the data is required. |
| sliceRecovery <n> | If the slice is in the mist of recovery where existing segments are written back to the slice. Either 1 for true or 0 for false. <n> denotes the index of the slice. |
| svrMapCount | Number of file slice server ID mappings. The mappings contain the location of the file slice store as well as credentials to the file slice store if required. Only mappings that are used by the slices are returned. |
| svrId<n> | ID of the file slice store. <n> denotes the index of the mapping. |
| svrHost<n> | URI of the file slice store. <n> denotes the index of the mapping. |
| svrType<n> | Type of the file slice store. Either 1 for standalone file slice store, or 2 for Windows Azure file slice store, or 3 for Amazon S3 file slice store. <n> denotes the index of the mapping. |
| svrMode<n> | Mode that is supported by the file slice store. Either 1 for streaming (for random read write, also support non-streaming), or 0 for non-streaming. <n> denotes the index of the mapping. |
| svrKey<n> | Secure random value to compute the request signature for standalone file slice store. <n> denotes the index of the mapping. |
| svrOpt<n>.<prop name> | Additional properties for the file slice store. If the file slice store is of Windows Azure or Amazon S3, then the properties would include the credentials to access the slice store. <n> denotes the index of the mapping. <prop name> denotes the property name. |

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Directory Response Body:  idaVersion=1  idaMatrix=1,2,3,4,5|6,7,8,9,10|11,12,13,14,15  idaShares=5  idaQuorum=3  type=1  owner=Victoria  creation=1282817514297  lastModified=1282817514297  lastAccessed=1282817514297  File Response Body:  idaVersion=1  idaMatrix=1,2,3,4,5|6,7,8,9,10|11,12,13,14,15  idaShares=5  idaQuorum=3  type=0  owner=Victoria  creation=1282817514297  lastModified=1282817514297  lastAccessed=1282817514297  size=150  blkSize=102400  keyHash=sd2sd5  fileMode=-1  sliceCount=2  sliceSeq0=1  sliceName0=slice1  sliceServerId0=tfs1  sliceLength0=20  sliceChecksum0=dfg445j645kjsfkyre230jkdfgcioe  sliceRecovery0=1  sliceSeq1=2  sliceName1=slice2  sliceServerId1=tfs2  sliceLength1=60  sliceChecksum1=j8374jsd873sdi2mim823f932ssdq  sliceRecovery1=0  svrMapCount=2  svrId0=tfs1  svrHost0=localhost\:8010  svrType0=1  svrMode0=1  svrKey0=DF\*&$#FW324  svrOpt0.slicestore.status.address=localhost\:8011  svrOpt0 slicestore.status.ssl=off  svrId1=tfs2  svrHost1= http\://blob.core.windows.net/  svrType1=2  svrMode1=1  svrKey1=  svrOpt1.azure.storageaccount= moeifazuretest  svrOpt1.azure.secretaccesskey= IOpp4021LWmjaBJXoLgNun6Okw\=\=  svrOpt1.azure.retrycnt=5  svrOpt1.azure.retryinterval=5  svrOpt1.azure.usedevelopment=0 |

# Access File

The **Access File** operation updates the last accessed date time of an existing file metadata record under the specified namespace.

Last accessed date time is usually updated when the file is opened for read/write.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/access

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| filename | Required. Full path, including the name of the file. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/access?namespace=urn:nyp.edu.sg&filename=%2Fsample.txt |

Response

The response includes an HTTP status.

**Status Code:**

A successful operation returns status code 200 (OK).

If the file path provided does not exist, status code 400 (Bad request) is returned.

If the path provided is not a file, status code 409 (Conflict) is returned.

For more information on other status codes, refer to Appendix.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK |

# Generate File Slice(s)

The **Generate File Slice(s)** operation creates new file slice information to be used by the client. The information includes the name of the slice, the file slice store to store the content of the file slice etc. The file slice store that is allocated is random.

The function is used when writing to a new file or when a segment is required by the client because the current slice store where the content of the slice is previously stored is not available.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/generate

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| size | Required. Number of slices required by the client. |
| mode | Required. Preferred mode of the file slice store where the content of the slice will be stored. Either 1 for streaming or 0 for non-streaming. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/generate?namespace=urn:nyp.edu.sg&size=2&mode=1 |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If no file slice store is registered under the specified namespace, status code 400 (Bad request) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The content is made up of name value pairs with each pair on a new line.

|  |  |
| --- | --- |
| Name | Description |
| sliceName<n> | Name of the file slice. <n> denotes the index of the slice. |
| sliceServerId<n> | ID of the file slice store that is allocated to the slice to store its content. <n> denotes the index of the slice. |
| svrMapCount | Number of file slice server ID mappings. The mappings contain the location of the file slice store as well as credentials to the file slice store if required. Only mappings that are used by the slices are returned. |
| svrId<n> | ID of the file slice store. <n> denotes the index of the mapping. |
| svrHost<n> | URI of the file slice store. <n> denotes the index of the mapping. |
| svrType<n> | Type of the file slice store. Either 1 for standalone file slice store, or 2 for Windows Azure file slice store, or 3 for Amazon S3 file slice store. <n> denotes the index of the mapping. |
| svrMode<n> | Mode that is supported by the file slice store. Either 1 for streaming (for random read write, also support non-streaming), or 0 for non-streaming. <n> denotes the index of the mapping. |
| svrKey<n> | Secure random value to compute the request signature for standalone file slice store. <n> denotes the index of the mapping. |
| svrOpt<n>.<prop name> | Additional properties for the file slice store. If the file slice store is of Windows Azure or Amazon S3, then the properties would include the credentials to access the slice store. <n> denotes the index of the mapping. <prop name> denotes the property name. |

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  sliceName0=slice1  sliceServerId0=tfs1  sliceName1=slice2  sliceServerId1=tfs2  svrMapCount=2  svrId0=tfs1  svrHost0=localhost\:8010  svrType0=1  svrMode0=1  svrKey0=DF\*&$#FW324  svrOpt0.slicestore.status.address=localhost\:8011  svrOpt0 slicestore.status.ssl=off  svrId1=tfs2  svrHost1= http\://blob.core.windows.net/  svrType1=2  svrMode1=1  svrKey1=  svrOpt1.azure.storageaccount= moeifazuretest  svrOpt1.azure.secretaccesskey= IOpp4021LWmjaBJXoLgNun6Okw\=\=  svrOpt1.azure.retrycnt=5  svrOpt1.azure.retryinterval=5  svrOpt1.azure.usedevelopment=0 |

# List File(s)

The **List File (s)** operation retrieves the immediate list of files/directory of a specified directory.

**Resource Type:** Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/list

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the directory belong to. |
| path | Required. Full path, including the name of the directory. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/list?namespace=urn:nyp.edu.sg&path=%2Fabc |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If no file slice store is registered under the specified namespace, status code 400 (Bad request) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The content is made up of the immediate list of files/directory with each record on a new line.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  /abc/def  /abc/sample.txt |

# Refresh Directory

The **Refresh Directory** operation checks if there are changes to the directory (E.g new file/directory added, file property like last updated date time changed etc) and returns the immediate list of files/directories. Brief information for each file/directory is returned as well.

**Resource Type:** Directory

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/refresh

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| path | Required. Full path, including the name of the directory. |
| lastChk | Required. The date time in milliseconds since January 1, 1970, 00:00:00 GMT when the function is last invoked. If the function is invoked for the first time, the value 0 can be used. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/refresh?namespace=urn:nyp.edu.sg&path=%2Fabc&lastChk=0 |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

If the path provided does not exist, status code 400 (Bad request) is returned.

If the path provided is not a directory, status code 409 (Not acceptable) is returned.

For more information on other status codes, refer to Appendix.

**Response Body:**

The content is made up of name value pairs with each pair on a new line.

|  |  |
| --- | --- |
| Name | Description |
| lastCheck | The date time in milliseconds since January 1, 1970, 00:00:00 GMT when the function is invoked. |
| fileCount | Number of files in the response body. -1 denotes that there are no changes to the files/directories under the specific directory. |
| fileName<n> | Full path of the file/directory. <n> denotes the index. |
| size<n> | Size of the file in bytes. If the record is a directory, the value will be 0. <n> denotes the index. |
| type<n> | Type of the record. Either 0 for file or 1 for directory. <n> denotes the index. |
| creation<n> | Creation date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT <n> denotes the index of the mapping. |
| lastModified<n> | Last updated date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT <n> denotes the index of the mapping. |
| lastAccessed<n> | Last accessed date time of the file/directory in milliseconds since January 1, 1970, 00:00:00 GMT <n> denotes the index of the mapping. |

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  lastCheck=1282817514297  fileCount=2  fileName0=/abc/def  size0=0  type0=1  creation0=1282817514297  lastModified0=1282817514297  lastAccessed0=1282817514297  fileName1=/abc/sample.txt  size1=150  type1=0  creation1=1282817514297  lastModified1=1282817514297  lastAccessed1=1282817514297 |

# Change Mode

The **Change Mode** operation transfer the file slices of existing file to file slice stores that support the given mode.

This operation can only be performed if the slices to be transferred are not in the mist of recovery.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/chgmode

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| by | Required. ID of the user attempting to obtain the lock. |
| filename | Required. Full path, including the name of the file to lock. |
| mode | Required. The mode of the file slice stored the transferred slices must support. Either 1 for streaming or 0 for non-streaming. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/lock?namespace=urn:nyp.edu.sg&filename=%2Fsample.txt&by=victoria&mode=1 |

Response

The response includes an HTTP status.

**Status Code:**

A successful operation returns status code 200 (OK).

If the path of the file provided does not exist or the path provided is not a file, status code 400 (Bad request) is returned.

If the file is currently open for write related operations by another user, status code 423 (Locked) is returned.

For more information on other status codes, refer to Appendix.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK |

# Check File Status

The **Check File Status** operation retrieves the status of file mode change or slice recovery.

**Resource Type:** File

Request

**URI:** http[s]://<host name or IP address>[:<port number>]/file/chkstatus

**Request Parameters:**

|  |  |
| --- | --- |
| Name | Description |
| namespace | Required. Namespace the file belong to. |
| filename | Required. Full path, including the name of the file to lock. |
| status | Required. The type of status to retrieve. Either “mode” for file mode change or “rev” for slice recovery. |

**Sample Request:**

|  |
| --- |
| Request Syntax:  http://localhost:9010/file/chkstatus?namespace=urn:nyp.edu.sg&filename=%2Fsample.txt&status=rev |

Response

The response includes an HTTP status and response body.

**Status Code:**

A successful operation returns status code 200 (OK).

For more information on other status codes, refer to Appendix.

**Response Body:**

The file mode change or slice recovery status.

**Sample Response:**

|  |
| --- |
| Response Status:  HTTP/1.1 200 OK  Response Body:  1 |

# Appendix

|  |  |
| --- | --- |
| Code | Description |
| 404 (Not found) | If the file/directory is not found. |
| 406 (Not acceptable) | Request parameter(s) is missing or invalid. |
| 500 (Internal server error) | An error that occurred when processing the request. Look at the message for more details. |