RSD

RSD FOLDERS
Java API

Version 5. 1. x

Reference Guide

www.rsd.com

Manual: RSD Folders Java API - Version 5.1

Contents

Concepts and Terminology

Section overview

This section introduces the RSD Folders Java API.

Topics covered in this section:

- Basic requirements - see below
- "New features and enhancements" on page 10
- "Methodology for retrieving information" on page 13
- "Exceptions and error messages" on page 14
- "General information on methods" on page 14
- "Diagram of objects" on page 15

Basic requirements

The RSD Folders Java API enables the retrieval of information stored in an RSD Folders database for use with any customized application.

Requirements for the RSD Folders Server

The RSD Folders server can reside in a z/OS, Unix or Windows server environment. The Global View Monitor can be defined in a z/OS environment.

Depending on the platform, communication with the RSD Folders server is effected via an Online Monitor (for z/OS) or an Online server (for Unix or Windows).

Requirements for the API application

The customized application can reside anywhere and should be written in Java.

The application requires Java version 1.2 or higher.

The Java API accomplishes its task by facilitating the creation of predefined objects to hold the information and by providing the functionality to execute actions to obtain the information.

Information on the requirements for using and the process for installing the RSD Folders Java API can be found in the readme.txt file on the installation CD-ROM.

The RSD Folders Java API is written in Java.

Requirements for the programmer

The programmer should have a good understanding of RSD Folders as presented in the documentation, "Management Guide & Technical Reference", accompanying the RSD Folders server product running on a z/OS, Unix or Windows server platform.

In addition, (s)he should have knowledge of programming in Java.

New features and enhancements

Version 5.1

Document history (new)

Support for viewing a history of actions performed on a document. For information on the objects and interfaces that implement this functionality, see "Manage document history" on page 82.

There are new methods to retrieve authorizations to use document history functionality. For more information, see "Get information on user authorizations" on page 27.

Folder transactions (enhanced)

A new object updates, initializes and terminates a transaction between the client application and an RSD Folders Server. For more information, see "Index Update transaction" on page 106.

Additionally a new implementation of the newIndexFiledUpdate method has been added. See "newIndexFieldUpdate" on page 65.

Version 5.0

Document versions (new)

Support for creating versions of a document. This includes functionality to check out and lock the latest version of a document, check in a new version and unlock the document, and add labels to a version. For information on the objects and interfaces that implement this functionality, see "Manage document versions" on page 76.

There are new methods to retrieve authorizations to use version functionality. For more information, see "Get information on user authorizations" on page 27.

In FLDDocument, there are two new methods, isLocked and hasVersions, to retrieve information on the document versioning status. For more information, see "General document information" on page 60.

Notes (new)

Support for associating notes with a document. This includes functionality to create, view, modify and delete text notes and attachments. For information on the objects and interfaces that implement this functionality, see "Manage notes" on page 71.

In addition, there are new methods to retrieve authorizations to use note functionality. For more information, see "Get information on user authorizations" on page 27.

Digital signature support (new)

The FLDDocumentInfo object contains the isDocumentSigned method to return information on the validity of a digital signature. For more information, see "Technical document information" on page 65.

Upload a file (enhanced)

There is a new method FLDUploadTransfer that manages the uploading of a document. For more information, see "Upload a file" on page 53.

Filter options (enhanced)

When setting a filter, It is now possible to include or exclude folders or documents that match the folder or document name criteria. For more information, see "Returning folders/documents that match or do not match the criteria" on page 39.

Version 4.5.1

FLDConnection (changed)

There is a new constructor that specifies the user name /password and server name/port.

There is a new method, setDataCharset, which replaces setEncodingName and setEncodingFile.

There is a new method, setForceIdentification, which can be used to handle logon problems, where the server finds that the logon name is already being used.

Explanation of using the document ID (new)

The document ID can be used to directly access document information and perform actions related to a document. For more information, see "Unique document ID functionality" on page 20.

FLDDocument (enhanced)

There are new methods for determining the format of a document: isMail, isMailAttachment, isPDF.

There are new methods to define the sort order for a document list. For more information, see "Define list sort order" on page 56.

List of documents returned by an ABI query (changed)

When an ABI query is made, an initial list of documents that match the query criteria is built on the server.

In previous versions, the result list actually returned was a subset of the initial list and included only documents that matched the active filter.

Now, result list returned is based on one of the following:

- The subset of documents that match the criteria set in the active F/D filter.
- The subset of documents that match the criteria set in all F/D filters defined by a list of F/D filters, which can designated by the user.

For more information, see "Query results returned by the server" on page 95

Version 4.4.x

The ability, once a connection is made, to immediately retrieve document information and manage documents. For more information, see "Unique document ID functionality" on page 20.

Version 4.2.1

The new functionality, available in the API, is dependent on the platform where the RSD Folders server is running and the software version. Normally, methods accessible in earlier versions remain available to support programs written using them. However, older methods that have been changed or moved to other objects or interfaces are noted in the Java doc as deprecated and are not documented here. Thus, if you cannot find a method in this version of the documentation (that you have been using), refer to the Java doc that comes with the API installation. It is suggested, whenever possible, to update your programs to use the new methods.

Interfaces (new)

To make it easier to reference specific information (for example a document), for many of the objects the information for a specific record is retrieved in an interface. Thus when navigating in a list, the information is retrieved using methods in an associated interface. This allows you, instead of supplying detailed information, to reference an instance of the interface that contains the information. Normally, the methods previously used to retrieve the information are available from the interface or from the object, though the new method is preferred.

Length of document and folder names (changed)

Versions less than 4.2 of RSD Folders support a maximum document name length or 40 characters and a maximum folder name length of 32 characters. Versions 4.2 or higher of RSD Folders support a maximum document name length or 250 characters and a maximum folder name length of 120 characters. For RSD Folder running on a Unix or Windows server platform, the length of the folder name can be specified (reduced) for each folder type. There are methods in the FLDConnection object to verify the maximum document and folder name size for the connected server.

Support for mark functionality (new)

When connected to an RSD Folders server version 4.2 or higher running on a Unix or Windows server platform where mark functionality is implemented, methods are available to manage and modify marks. Objects and interfaces where mark functionality is referenced:

- The FLDMarkList object lists the marks defined on the server. It also returns the logged-on user's authorizations for viewing and changing the mark status. See "Retrieve a list of mark types" on page 36.
- The FLDSetFilter object supports filtering a list on mark status. See "Set a filter" on page 38.
- The FLDDocument interface contains methods to view and change the status of a mark. See "General document information" on page 60.

Support for relative date (new)

When connected to any RSD Folders server version 4.2 or higher, when specifying a date in the FLDSetFilter object, the date can be specified as relative. When a relative date is defined, then the date is dynamically created relative to the current computer date. Absolute date functionality, specifying a date value, is still supported. For more information, see "Date criteria" on page 43.

Support for changing the value of an index entry (new)

When connected to an RSD Folders server version 4.2 or higher running on a Unix or Windows server platform where Advanced Business Index (ABI) functionality is implemented, the index values can be modified. For more information, see "Add, delete or change index entries" on page 103.

Support for retrieval of index values from a document list (new)

When connected to any RSD Folders server where ABI functionality is implemented, the index values for a document can be returned from a document list. For more information, see "getDocumentIndexes" on page 62.

Methodology for retrieving information

To retrieve information from an RSD Folders server database or a Global View Monitor, communication must be established which supplies, among other information, a user name.

The API communicates the logged-on user name to the server and, based on the filters and authorizations for that logged-on user, provides the available information that matches the filter/authorization criteria.

Thus the user:

- must be defined.
- must be assigned, one or more filters and appropriate authorizations.

The programmer uses the built-in functionality to automate and streamline the process of retrieving the necessary data.

Filters

Filters are created within an RSD Folders server database. They contain criteria that are used to limit the folders and documents available when that filter is set.

For access to information, it is required that each logged-on user name logging on to an RSD Folders server be assigned one or more filters.

For folder types and document types, the related filter definition arguments can either specify a specific type or can be left blank (which implies access to all defined types).

Authorizations

Authorizations are required for each logged-on user name logging on to an RSD Folders server. Among other things, they can further limit which folder types and document types the logged-on user can access.

For both folder types and document types, each authorization definition can specify a range of both folder types and document types. It is assumed that this range falls within scope defined in the filters assigned to the logged-on user.

In addition, each user is assigned authorizations to perform specific functionality. Thus a user can only rename a folder if, on the RSD Folders server accessed, the user has been assigned this authorization.

Example

Definitions in the RSD Folders database being accessed:

- Folder types 1-7 are defined.
- A filter ALL is defined with the folder type restriction blank (access to all folder types). This is the filter that is assigned to all logged-on users.
- Logged-on user AAAA is authorized to access folder types 2, 4-6.
- Logged-on user BBBB is authorized to access folder types 1-4.

Default folder type and valid folder types

The intersection of the active filter (the folder and document definitions) and the logged-on user's authorizations define:

- The range of *valid folder types*.
  - For logged-on user AAAA, the valid folder types would be 2, 4, 5 and 6.
  - For logged-on user BBBB, the valid folder types would be 1, 2, 3 and 4.
- The *default folder type*, which is always the first numerically accessible folder type in the range of valid folder types.
  - For logged-on user AAAA, the default folder type would be 2.
  - For logged-on user BBBB, the default folder type would be 1.

If the folder type is modified in the active filter, or in the case of a folder list, the folder list accessed is not of the default folder type, then the "changed" folder type must be within the range of valid folders types.

General information on methods

Generally, the methods described below are available for most objects.

The methods for each object are normally grouped first by functionality, with set methods presented before get methods and then within each functional grouping, methods are normally presented in alphabetic order.

Close method

The "close" method notifies the server to stop processing for this object. It is required for most objects, except those noted below, and is therefore is only mentioned here.

Method: close()

Returns: void.

Objects that do not have a close method: FLDCreateFolder, FLDDeleteDocument, FLDDescriptorList, FLDDocumentInfo, FLDException, FLDExceptionMsg, FLDLanguage, FLDTextDocumentFind, Version.

Set methods

Most objects use 'set' methods. These methods either set required values or restrict the information that is returned to the object. Normally, but not always, 'set' methods should be used before 'read' or 'execute' methods.

Except where noted, set methods return: void.

Navigation methods

These methods obtain a list of records (or data blocks).

Navigate forward

A 'readFirst' method fills a buffer with records and positions at the first record in the buffer. A 'readNext', 'locateFirst', 'locateNext' (or other similar positioning methods) fills the buffer with records starting with the record specified (or if not found, with the next record), positioned at the first record in the buffer. After one of the above methods has been executed, information can be retrieved. Additional navigation methods move to the next record.

Navigate backward

A 'readLast' method fills a buffer with records starting from the end of the records and positions at the last record in the buffer. A 'readPrevious' or 'locatePrevious' (or other similar positioning methods) fills the buffer with records ending with the record specified (or if not found, with the previous record), positioned at the last record in the buffer. After one of the above methods has been executed, information can be retrieved. Additional navigation methods move to the previous record.

Retrieving information

Most objects contain 'get', 'has' and 'is' methods ' to retrieve information.

Exceptions and error messages

Exceptions

A double asterisk ** after a method or constructor indicates that the execution of the method or constructor can generate an exception. When an exception occurs, an error message is generated.

Examples: readFirst() ** and FLDConnection() **

Exceptions can be generated when a method is executed.

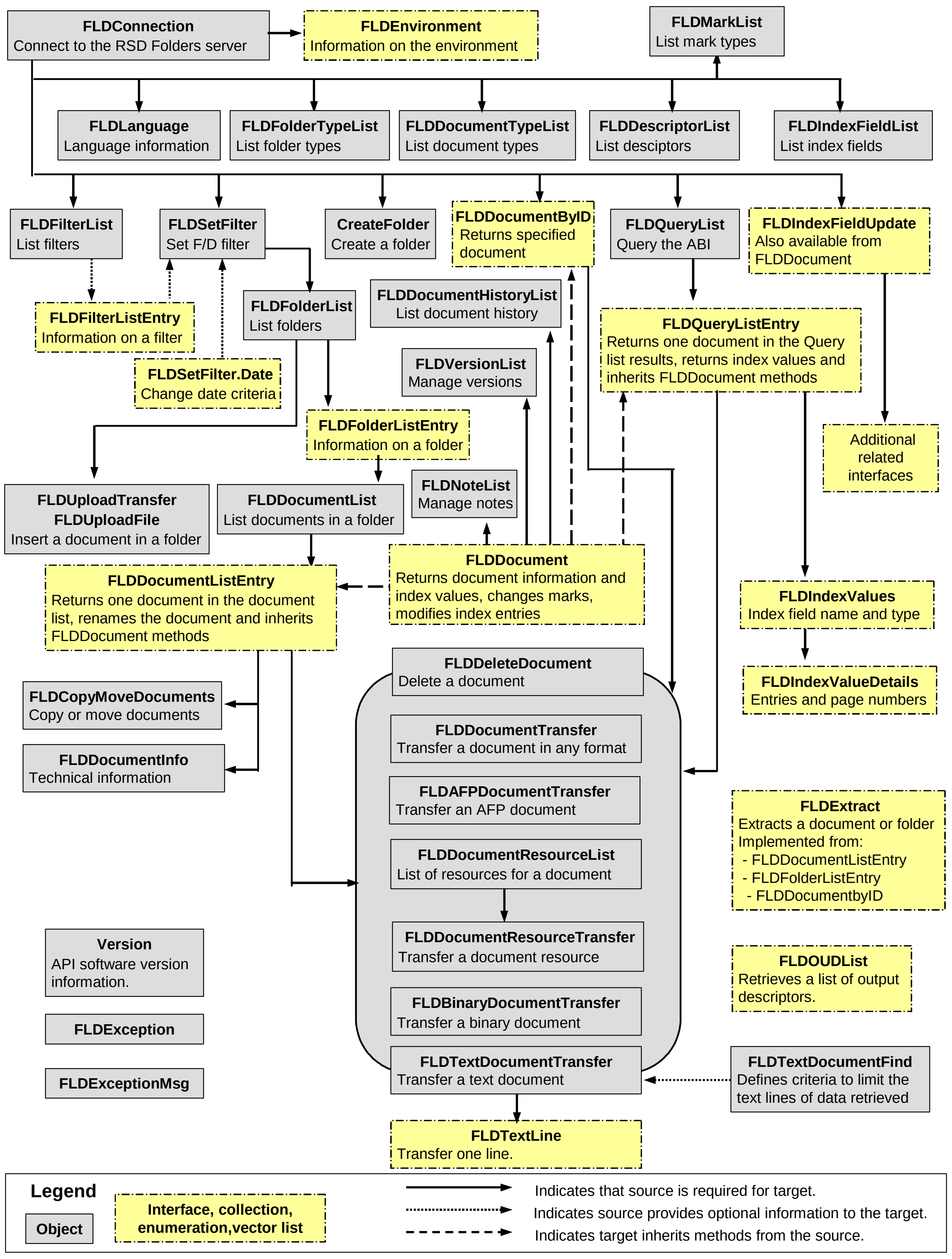
For example, the connect method, contained in for the FLDConnection object, can generate an exception if an error occurs during the connection with the server.

Error messages

For more information on accessing error messages and for a list of the error messages returned, see "Error messages" on page 125.

Diagram of objects

The following diagrams show the general relationship of objects in the API.



Generic example of retrieving information

The procedure below shows some of the basic steps involved in using the API: making a connection, setting a filter, accessing a folder list, accessing a document list and closing objects.

```
// Connection
connect = new FLDConnection();
connect.setAddress(IPaddress);
connect.setPort(port);
connect.setUserID(userID);
connect.setPassword(password);
connect.connect();

// FLDSetFilter and FLDFolderList
FLDSetFilter mySetFilter = new FLDSetFilter(connect);
String filterName = "myFilter";
mysetFilter.applyFilter(filterName);

FLDFolderList folderList = new FLDFolderList(mySetFilter);
boolean hasNextFolder = folderList.readFirst();
while(hasNextFolder){
        FLDFolderListEntry folderListEntry = folderList.getFolderListEntry();
        String folderName = folderListEntry.getFolderName();
        int folderType = folderListEntry.getFolderType();

        // update your boolean to check if there is another entry in the list
        hasNextFolder = foldList.readNext();
}

// FLDDocumentList
FLDDocumentList documentList = new FLDDocumentList(folderList);
boolean hasNextDoc = documentList.readFirst(folderName, 1);
while(hasNextDoc){
        FLDDocumentListEntry documentListEntry = documentList.getDocumentListEntry();
        String docName = documentListEntry.getDocumentName();
        int docNbrPages = documentListEntry.getDocumentNumberPages();

        // update your boolean to check if there is another entry in the list
        hasNextDoc = documentList.readNext();
}

// Close all objects not needed anymore (reverse order as that of creation)
documentList.close();
folderList.close();
setFilter.close();
connect.close();
```

Server connection object

Section overview

This section covers connecting to an RSD Folders Server, retrieving version information and setting the language for error messages.

Topics covered:

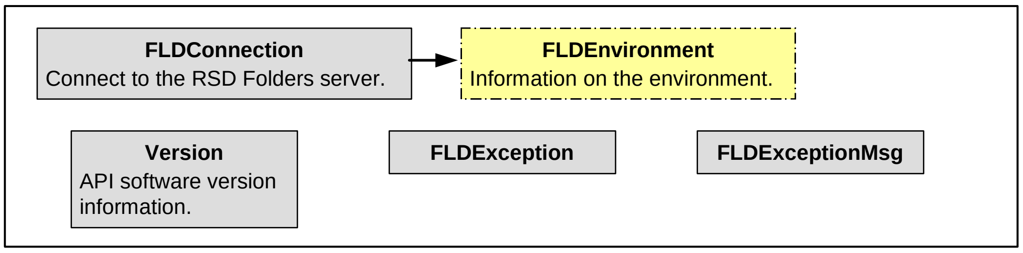
- "Initiate server/client connection" on page 18.

  The FLDConnection object makes the connection to the server and permits the selection of an environment (if more than one exists for that connection). It is required.

- "Version information" on page 29.

  The Version object returns information on the version number.

Objects covered in this section



FLDConnection
Connect to the RSD Folders server.
FLDEnvironment
Information on the environment.
Version
API software version information.
FLDException
FLDExceptionMsg

For information on FLDException and FLDExceptionMsg, see "Error messages" on page 125.

Initiate server/client connection

The **FLDConnection** object initializes and terminates communication between the client application and an RSD Folders Server.

Constructor: FLDConnection() **

Constructor: FLDConnection(String userID, String password, String address, int port) **

For more information on these parameters, see corresponding methods under "Initialization methods" on page 22.

- "userID" should be the name of a super user. For more information, see "Methods for connecting to the server" on page 19.
- "password" is the password for the user.
- "address" is the online monitor/server address.
- "port" is the online monitor/server port number.

Important - please read these topics before initiating a connection to the server.

- "Methods for connecting to the server" on page 19 - covers types of connections that can be initiated.
- "Unique document ID functionality" on page 20 - covers requirements for using the document id for quick access to document information and functionality.
- "Connection example" on page 21 - shows example code for initiating a connection.

Methods available for this object:

- "Initialization methods" on page 22
- "Establish the connection" on page 23
- "List and set environments" on page 23
- "Add, delete or update an index entry value" on page 28
- "Get information on the connection" on page 24
- "Get information on user authorizations" on page 27

Methods for connecting to the server

The connection to the server is always made for a single user, identified throughout the documentation as the logged-on user. This user must have all authorizations required to perform any tasks executed.

There are two possible connection types that can be set, one that requires the password for the logged-on user and the second which avoids the need to know that password when changing logged-on users during a session.

The setIdentificationType method defines the connection type:

- IDENTIFICATION_TYPE_USER, the default, uses the UserID as the logged-on user.
- IDENTIFICATION_TYPE_SERVER, uses the EndUserID as the logged-on user and avoids the need to enter the password for that user. This connection type is required if the purchase of the API software includes a provision that limits the number of logged-on users.

To initialize the connection using IDENTIFICATION_TYPE_USER (the default):

- Execute the setUserID method and, if required at your site, the setPassword method.
- The user defined in the setUserID method is considered the logged-on user.

To initialize the connection using IDENTIFICATION_TYPE_SERVER:

- Execute the setIdentificationType, initialized with the constant IDENTIFICATION_TYPE_SERVER.
- Execute the methods setUserID and setPassword, with the name and password for a "super" user (defined with all authorizations).
- Execute the method setEndUserID, with the name of the user for whom the tasks will be performed. The name used here is considered the logged-on user.

Notes that the UserID and the EndUserID parameters:

- Both must be user names defined in the RSD Folders database being accessed. For server version 4.1 or higher, the maximum length for the user name is 255 characters. For server version 4.0, the maximum length is 8 characters.
- The UserID should be a "super" user assigned all filters and all authorizations that are necessary for all of the EndUserIDs for whom tasks will be performed during the session. It requires the UserID password.
- The EndUserID should be the user name for whom actual tasks are performed. This user is referred to throughout the documentation as the "logged-on user". No password is required.
- If no EndUserID is provided, it is assumed that tasks should be performed using the UserID. The user name provided for this UserId is then considered as the logged-on user.

Unique document ID functionality

When unique document ID functionality is activated, document information can be retrieved and actions (such as transferring a document) can be performed without first going through the process of setting a filter, choosing a folder list, etc.

To use this functionality, the document ID for the document must be known. This information is returned by the following methods:

- In FLDUploadFile: getDocumentID, getRawDocumentID. After a document has been uploaded, these methods return the unique document id assigned by the server. For more information, see "Upload a file" on page 53.
- In FLDDocument: getDocId, getRawDocID. For more information, see "General document information" on page 60.

  Note, these methods only return information if either:

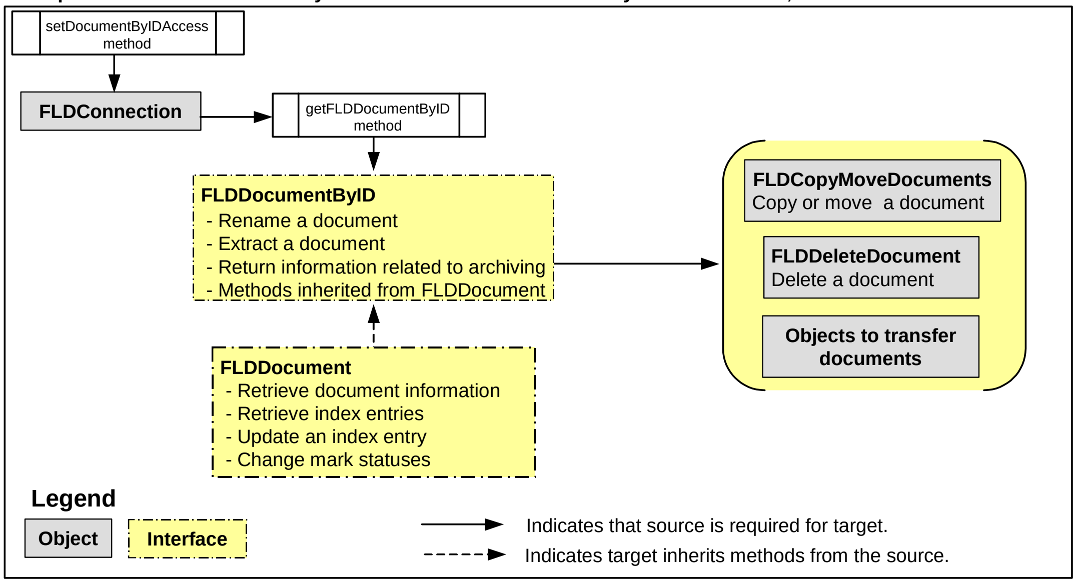
  - In FLDQueryList, the method "enableDocumentID", documented on page 99, is executed, set to True.
  - In FLDDocumentList, the method "setDocumentIDExtensionInfo", documented on page 56, is executed, set to True.

Requirements for using unique document ID functionality:

1. The functionality to generate unique document ids must be activated on the server being accessed.
2. Before making a connection to the server, you must execute the method "setDocumentIDAccess" documented on page 22 (set to True). When executed, the API queries the server to find out if unique document id functionality is activated on the server. If it is not, an exception will be generated.
3. Once the connection is made, you must execute the method "getFLDDocumentByID" documented on page 25. The method implements an instance of the FLDDocumentByID interface, which allows direct access to document information and functionality. For more information, see "Direct access to document information" on page 67.

Once the requirements have been met, the following document ID functionality is available.

Unique document id functionality - note that all other functionality is also available, but not shown here


setDocumentByIDAccess method
FLDConnection
getFLDDocumentByID method
FLDDocumentByID
- Rename a document
- Extract a document
- Return information related to archiving
- Methods inherited from FLDDocument
FLDDocument
- Retrieve document information
- Retrieve index entries
- Update an index entry
- Change mark statuses
FLDCopyMoveDocuments
Copy or move a document
FLDDeleteDocument
Delete a document
Objects to transfer documents
Legend
Object
Interface
Indicates that source is required for target.
Indicates target inherits methods from the source.

Important

- When using unique dcoument id functionality, the API functionality is not limited to the objects shown above. All API functionality is always available.
- As with all API functionality, when using the document id, the logged-on user must be authorized to access the document. If performing an action on a document, (s)he must be authorized to perform the action.

Connection example

To make a connection to the server:

1 Create the FLDConnection object.

Example: `FLDConnection connectInfo = new FLDConnection();`

2 Initialize the object with the following values using the method shown in parentheses.

- The IP host address (setAddress)
- The port number (setPort)
- The super user id (setUserId)
- The super user password (setPassword)

Optionally, if necessary

- To have a single connection that permits changing the end user id without having to specify an end user password, set the identification type to IDENTIFICATION_TYPE_SERVER and specify an end user id. For more information, see "Methods for connecting to the server" on page 19.
- Set the language for error messages. For more information, see "setLanguage" on page 22.
- The translation method for converting between ASCII and EBCDIC. For more information, see "setDataCharset" on page 22.
- Specify access to the document ID, for direct access to a document. For more information, see "setDocumentIDAccess" on page 22.

In the examples below, all variables have been predefined.

Example of simple logon for one user:

```
connectInfo.setAddress(IPAddress);
connectInfo.setPort(port);
connectInfo.setUserID(userId);
connectInfo.setPassword(password);
connectInfo.setDataCharset(myCharSet);
connectInfo.setDocumentIDAccess(docIDAccess); // to use document id
functionality
```

Example of connection that permits changing the end user id without specifying a password:

```
connectInfo.setAddress(IPAddress);
connectInfo.setPort(port);
connectInfo.setIdentificationType(FLDConnection.IDENTIFICATION_TYPE_SERVER);
connectInfo.setUserID(userId);
connectInfo.setPassword(password);
connectInfo.setEndUserID(enduserId);
connectInfo.setDataCharset(myCharSet);
connectInfo.setDocumentIDAccess(docIDAccess); // to use document id
functionality
```

3 Establish the connection.

Example: `connectInfo.connect();`

4 If necessary, find and set the required environment.

5 Perform the tasks necessary for the logged-on logged-on user.

6 When all tasks have been performed, close all open objects.

7 Close the FLDConnection object.

Example: `connectInfo.close();`

Initialization methods

setAddress (required)

Method: setAddress(String address)

Parameter: "address" is the RSD Folders Online Monitor/Server IP address. For example, "1.2.1.2".

setDataCharset (optional)

Data is sent from the RSD Folders Online Monitor/Server to the API Client application, in EBCDIC. To correctly display the data in a browser, the EBCDIC characters must be translated to ASCII. This method can be used to specify the character set to us for the conversion.

Method: setDataCharset(String charsetName) **

Parameter: "charsetName " is a standard character set for performing ASCII/EBCDIC conversion. Default: cp500. The character set should match the source data sent by the server. Some examples:

- cp500 - generic character translation
- cp273 - character translation supporting German character set
- cp297 - character translation supporting French character set

setDocumentIDAccess (optional)

This method must be executed (set to True) in order to use document ID functionality. For more information on using the document id, see "Unique document ID functionality" on page 20.

Method: setDocumentIDAccess(boolean docIDAccess) **

Parameter "docIDAccess" when set to:

- True, the API queries the server to find out if unique document id functionality is activated on the server. If it is not, you will receive an error message.
- False, the document ID cannot be used to directly retrieve information. This is the default.

setEndUserId (optional)

Method: setEndUserId(String endUserID) **

Parameter: "endUserID" is the user name of a logged-on user for whom tasks are performed.

setForceIdentification (optional)

If an attempt is made to logon and the error message # 241022 (user is already connected) is received, this method can be used to handle the problem. After executing this method, the connect method must again be executed.

Method: setForceIdentification(boolean forceIdentificationi)

Parameter: "force identification" if set to True, then the user is automatically logged off the first session and logged on, instead to the current session. If set to False, the logon with that user name is refused.

setIdentificationType (optional)

For more information on these constants and the purpose of this method, see "Methods for connecting to the server" on page 19.

Method: setIdentificationType(String identType)

Parameter: "identType" indicates whether the connection is initialized to allow changing the end user (setEndUserID method).

Permissible values: IDENTIFICATION_TYPE_SERVER, IDENTIFICATION_TYPE_USER.

Default: IDENTIFICATION_TYPE_USER.

setLanguage (optional)

Sets the language to be used for all product messages.

Method: setLanguage(FLDLanguage lang)

Parameter: "lang" is an instance that implements the FLDLanguage interface. For more information, see "Retrieve language used for error messages" on page 32.

Permissible values: ENGLISH, FRENCH, GERMAN. Default: ENGLISH.

setNewPassword (optional)

Method: setNewPassword(String newPassword) **

Parameter: "newPassword" is the new password for the user defined by the setUserID method.

setPassword (required)

Method: setPassword(String password) **

Parameter: "password" is the password for the user defined by the setUserID method.

setPort (required)

Method: setPort(int port)

Parameter: "port" is the port number of the RSD Folders Online Monitor/Server.

setResourcesPath

Method: setResourcesPath(String path)

Parameter: "path" is the path under which AFP resources are stored when a request is made to transfer an AFP document using an instance of the FLDDocumentTransfer object. If no path is set, the resources are transferred, but not stored.

setSecurityGroupName (optional)

This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.

Method: setSecurityGroupName(String securityGroupName) **

Parameter: "securityGroupName" is the name of a security group, maximum length of 8 characters.

setUserId (required)

Method: setUserId(String userID) **

Parameter: "userID" is the user name of the "super" user.

Establish the connection

Once the FLDConnection object has been created and initialized with the required values, this method must be used to make the connection to the server.

connect

Method: connect() **

Returns: void

List and set environments

The information on environments is useful mainly when connected to a Global View Monitor.

environments

Method: environments()

Returns: Enumeration. A list of valid environments for the logged-on user. The FLDEnvironment interface contains additional methods to retrieve environment names and other environment information.

Example: To print all environments (and additional information) for the connection connectInfo:

```
 for (Enumeration e = connectInfo.environments() ; e.hasMoreElements() ;)
{
 FLDEnvironment env = (FLDEnvironment)e.nextElement();
 System.out.println( env.getName() );
 System.out.println( env. getMaxDocLen () );
 }
```

Additional methods in the FLDEnvironment interface

- Method: activate() **

  Returns: void
- Method: getMaxDocLen()

  Returns: int. The maximum document name length defined on the server.
- Method: getMaxFldLen()

  Returns: int. The maximum folder name length defined on the server.
- Method: getName()

  Returns: String. The name of the environment.
- Method: getType()

  Returns: String. Indicates the type of RSD database. "F" indicates RSD Folders; "E" Indicates EOS. The RSD Folders Java API can only be used with an RSD Folders database.
- Method: isActivated()

  Returns: boolean. True, the environment is active. False, otherwise.

Get information on the connection

Once a connection has been established, the following methods return information on the connection. Note that the following methods are for internal use: getClientType, getCommunicationVersion.

getAddress

Method: getAddress()

Returns: String. The server IP address. For example, "1.2.1.2".

getDaysToProductExpiration

Method: getDaysToProductExpiration()

Returns: int. The number of days until the server product expires.

getEndUserId

Method: getEndUserId()

Returns: String. User name of the logged-on user, set using the setEndUserID method.

getFilterName

Method: getFilterName()

Returns: String. The default filter name for the logged-on user.

Note: getFLDDocumentByID found on next page - out of alphabetic order!

getFLDExceptionMsg

Method: getFLDExceptionMsg(FLDException e)

Returns: FLDExceptionMsg object. For more information and an example, see "Objects providing information on exceptions" on page 125.

getIdentificationType

Method: getIdentificationType()

Returns: String. The identification type set. For more information, see "Methods for connecting to the server" on page 19.

getLanguage

Method: getLanguage()

Returns: An instance of the FLDLanguage object. For more information, see "Retrieve language used for error messages on page 32.

getFLDDocumentByID

This method is only available if the method "setDocumentIDAccess", documented on page 22, has been executed. It allows direct access to document ID functionality.

Methods:

- getFLDDocumentByID(byte[] docID)
- getFLDDocumentByID(String docID)
- getFLDDocumentByID(byte[] docID, boolean enableDocumentDetails, boolean enableRunInfo)
- getFLDDocumentByID(String docID, boolean enableDocumentDetails, boolean enableRunInfo)

Parameters:

- "docID" is the document key returned by one of the following methods:
  - From FLDUpload: getDocumentID (returns String]) and getRawDocumentID (returns byte[]). See "Upload a file" on page 53.
  - From FLDDocumentInfo: getDocID (returns String]) and getRawDocID (returns byte[]). See "Technical document information" on page 65.
  - From FLDDocument: getDocID (returns String]) and getRawDocID (returns byte[]). See "General document information" on page 60.
- "enableDocumentDetails" defines what document information can be retrieved.
  - Default: False. Permissible values: True, False.
  - Background: An instance of the FLDDocument interface is used to return information on the document. Normally, only a subset of the methods are available. When this parameter is set to True, more methods available. For the information returned, see "General document information" on page 60.
- "enableRunInfo" defines whether the methods getRunDate and getRunNumber (page 63) can be executed to return information. Default: False. Permissible values: True, False.

Returns: An instance of the FLDDocumentByID interface. For more information, see "Direct access to document information" on page 67.

getMaxDocNameLength

Method: getMaxDocNameLength()

Returns: int. The maximum document name length defined on the server.

getMaxFolderNameLength

Method: getMaxFolderNameLength()

Returns: int. The maximum folder name length defined on the server.

getPort

Method: getPort()

Returns: int. The port number.

getResourcesPath

Method: getResourcesPath()

Returns: String. The path under which AFP resources are stored when a request is made to transfer an AFP document using an instance of the FLDDocumentTransfer object.

getSecurityGroupName

Method: getSecurityGroupName()

Returns: String. The security group name. This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.

getServerVersion

Method: getServerVersion()

Returns: String. The software version of the server.

getUserId

Method: getUserId()

Returns: String. User name of the "super" user, set using the setUserID method.

isEndUserIdentification

Method: isEndUserIdentification()

Returns: boolean. True, if server identification set to IDENTIFICATION_TYPE_USER. False, otherwise.

isServerTypeIdentification

Method: isServerTypeIdentification()

Returns: boolean. True, if server identification set to IDENTIFICATION_TYPE_SERVER. False, otherwise.

isSupported

Method: isSupported(int functionality)

Parameter: "functionality" permits verification of certain functionality on the server.

Permissible value	Server functionality supported when method returns True
ABI_MULTI_FILTERING_SUPPORTED	Specifying multiple filters when executing an ABI query.
DOCLIST_SORT_SUPPORTED	Sorting document lists.
EXCLUSIVE_FILTERING_SUPPORTED	When defining an filter that defines document and/or folder criteria, retrieving either documents/folders that match the criteria or those that do not match the criteria.
LONG_IDENTIFICATION_SUPPORTED	Specifying user names longer than 8 characters.
MARK_DOCUMENT_SUPPORTED	Mark status in documents.
RELATIVEDATE_FILTER_SUPPORTED	Relative date in a filter.
SIMPLE_AFP_DOCUMENT_SUPPORTED	transforming the data contained in an AFP document.
DOCUMENT_HISTORY_SUPPORTED	Specifying that document history is supported.

Returns: boolean. True, if the functionality is supported on the server. False, otherwise.

iszOSServer

Method: iszOSServer()

Returns: boolean. True, if connected to an RSD Folders server running on a z/OS platform. False, if connected to an RSD Folders server running on a Unix or Windows platform.

Get information on user authorizations

The following methods return information on the logged-on user authorizations defined on the RSD Folders server. For the methods listed, a mark in the:

- z/OS column indicates that the authorization can be defined on a z/OS platform.
- Unix/Windows column indicates the authorization can be defined on a Unix or Windows platform.
- API column indicates that the authorization is required for specific API functionality. When required for an object, it is noted. Other authorizations can be used to fine-tune the API application.

All methods return: boolean. True, if the user is authorized. False, otherwise.

Method name	z/OS	Unix/Windows	API
isApplyFilterToFolderModifyAuthorized	X	X	X
isDocCreateAuthorized()	X	X	X
isDocDeleteAuthorized()	X	X	X
isDocExportAuthorized()	X	X	
isDocMailAuthorized()	X	X	
isDocMoveCopyAuthorized()	X	X	X
isDocPCPrintAuthorized()	X	X	
isDocRenameAuthorized()	X	X	X
isDocViewAuthorized()	X	always authorized	X
isExtractAuthorized()	X		
isExtractDocAuthorized()	X		
isExtractDynamicAuthorized()	X		
isExtractFolderAuthorized()	X		
isExtractSingleDocAuthorized()	X	X	
isFilterDateModifyAuthorized()	X	X	X
isFilterMarkModifyAuthorized()		X	X
isFolderCreateAuthorized()	X	X	X
isFolderDeleteAuthorized()	X	X	
isFolderRenameAuthorized()	X	X	X
isGlobalQueryAuthorized		X	X
isIndexAddAuthorized()		X	X
isIndexDeleteAuthorized()		X	X
isIndexModifyAuthorized()		X	X
isIndexSearchAuthorized()		X	X
isIndexValueViewAuthorized()		X	X
isLocalAssociateFoldersAuthorized()	X	X	
isLocalDocCreateAuthorized()	X	X	
isLocalDocDeleteAuthorized()	X	X	
isLocalDocMoveCopyAuthorized()	X	X	
isLocalDocRenameAuthorized()	X	X	
isLocalFolderCreateAuthorized()	X	X	
isLocalFolderDeleteAuthorized()	X	X	
isLocalFolderRenameAuthorized()	X	X	
isLocalPCMgmtAuthorized()	X	X	
isLocalRemoteViewAuthorized()	X	X	
isNoteAddAuthorized()		X	X

Method name	z/OS	Unix/Windows	API
isNoteDeleteAuthorized()		X	X
isNoteUpdateAuthorized()		X	X
isNoteViewAuthorized()		X	X
isSchedulerAuthorized()		X	
isScheduleTransferAuthorized()	X	X	
isUpdatePrintCharAuthorized()	X	X	
isVersionAddAuthorized		X	X
isVersionDeleteAuthorized		X	X
isVersionPromoteAuthorized			
isVersionViewAuthorized		X	X
isPrivateDocumentHistoryAuthorized()		X	X
isPublicDocumentHistoryAuthorized()		X	X

Add, delete or update an index entry value

The following method is also available from an instance of the FLDDocumentListEntry or the FLDQueryListEntry interfaces.

newIndexFieldUpdate

Method: newIndexFieldUpdate() **

Result: Implements an instance of the FLDIndexFieldUpdate interface. For more information, see "Add, delete or change index entries" on page 103.

Version information

The **Version** object provides version information on the FLD Java API product.

Constructor: Version()

Methods

Note that the following method is for internal use only: toString.

getBuildDate

There are three methods that return the build date as a String.

The difference between the methods is the format used to return the date.

Method: getBuildDate()

Method: getBuildDate(java.util.Locale aLocale)

Parameter: "aLocale" defines the local format.

Method: getBuildDate(java.text.SimpleDateFormat format)

Parameter: "format" is a user defined format.

getBuildVersion

Method: getBuildVersion()

Returns: String. The product build version.

getImplementationVersion

Method: getImplementationVersion()

Returns: String. The product implementation version.

getProductTitle

Method: getProductTitle()

Returns: String. The product name.

getProductVendor

Method: getProductVendor()

Returns: String. The name of the company producing the product.

getSpecificationVersion

Method: getSpecificationVersion()

Returns: String. The product specification version.

getVersion

Method: getVersion()

Returns: String. The complete product version.

Initially available objects

Section overview

Below is a list of the objects that can be used once a connection to an RSD Folders Online Monitor/Server or Global View Monitor has been made. Many of these objects provide information that is used to set the parameter values for other constructors or methods.

Objects covered in this section

- "Retrieve language used for error messages" on page 32 (FLDLanguage object)

Objects covered in Section 4: Folder and filter objects

- "Retrieve a list of folder types" on page 34 (FLDFolderTypeList object)
- "Retrieve a list of document types" on page 35 (FLDDocumentTypeList object)
- "Retrieve a list of mark types" on page 36 (FLDMarkList object)
- "Create a folder" on page 37 (FLDCreateFolder object)
- "Retrieve a list of filters" on page 37 (FLDFilterList object)
- "Set a filter" on page 38 (FLDSetFilter object)

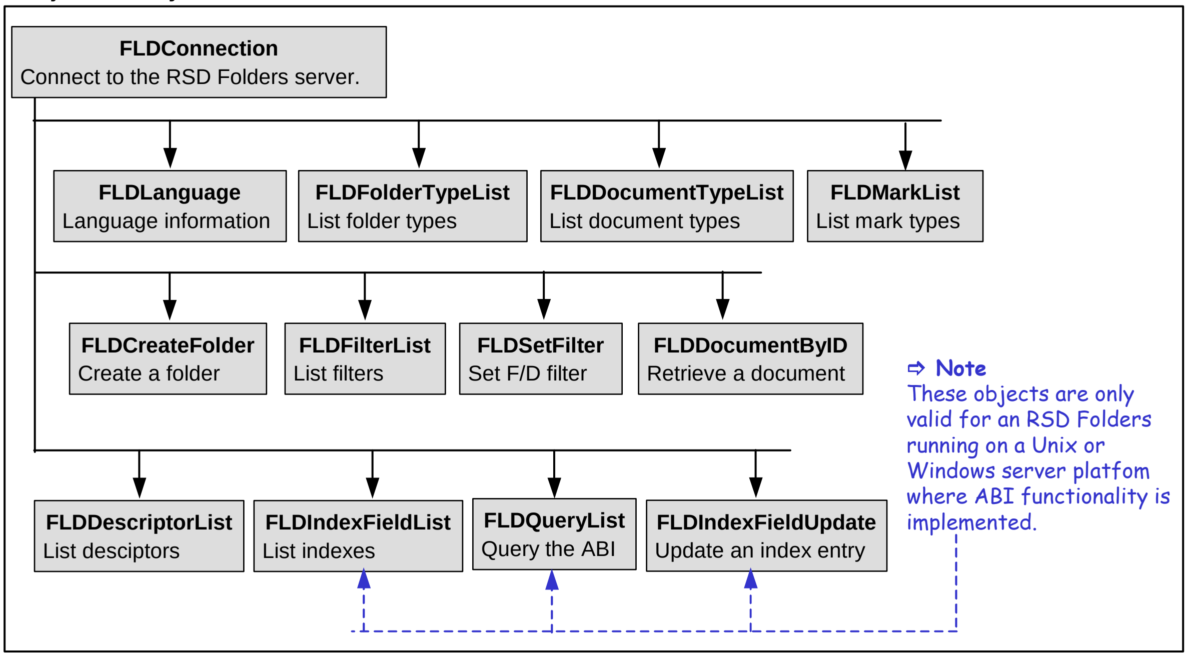
Objects covered in Section 5: Document objects

- "Retrieve a descriptor list" on page 52 (FLDDescriptorList object)
- "Direct access to document information" on page 67 (FLDDocumentByID object)

Objects covered in Section 6: Query objects and interfaces

- "Retrieve a list of index field descriptors" on page 92 (FLDIndexFieldList object)
- "Retrieve a list of documents based on an index query" on page 95 (FLDQueryList object)
- "Add, delete or change index entries" on page 103 (FLDIndexFieldUpdate object)

Objects initially available when there is a valid connection



Retrieve language used for error messages

The **FLDLanguag**e object returns information on the language used when retrieving error messages.

It is accessed using the getLanguage method (FLDConnection). For more information, see "getLanguage" on page 24.

Methods

Note that the following method is for internal use: getString.

getFLDExceptionMsg

Method: getFLDExceptionMsg(FLDException e)

Returns: FLDExceptionMsg object. For more information and an example, see "Objects providing information on exceptions" on page 125.

getLang

Method: getLang()

Returns: String. The language used to return error messages.

getLanguage

Method: getLanguage()

Returns: java.util.Locale. The current value of the FLDLanguage for this instance Locale.

getPropertiesLang

Method: getPropertiesLang()

Returns: String. The lowercase two character ISO-639 code.

Folder and filter objects

Section overview

Below is a list of the objects used to create a new folder, list filters, set the filter for a folder list and to retrieve a list of folders matching the filtering criteria.

Objects covered in this section

- "Retrieve a list of folder types" on page 34

  The FLDFolderTypeList object requires a connection to the server. It returns a list of valid folder types for the logged-on user.
- "Retrieve a list of document types" on page 35

  The FLDDocumentTypeList object requires a connection to the server. It returns a list of document types defined on the server.
- "Retrieve a list of mark types" on page 36

  The FLDMarkList object requires a connection to the server. It returns a list of mark types defined on the server and indicates, for each mark type, associated authorizations for the logged-on user.
- "Create a folder" on page 37

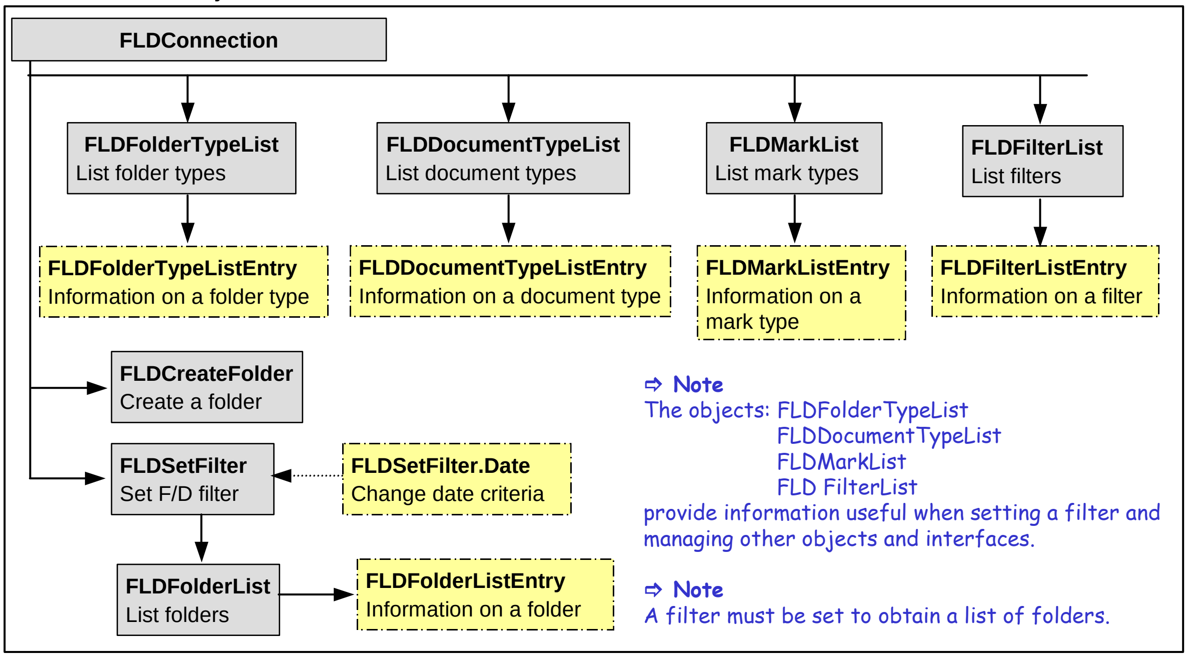
  The FLDCreateFolder object requires a connection to the server.
- "Retrieve a list of filters" on page 37

  The FLDFilterList object requires a connection to the server. It provides information that can be used to set a filter.
- "Set a filter" on page 38

  The FLDSetFilter object requires a connection to the server. It is required to retrieve a folder list.
- "Retrieve a list of folders" on page 47

  The FLDFolderList object requires a valid instance of the FLDSetFilter object. A valid instance of the FLDFolderList permits access to a document list.

Filter and folder objects



Retrieve a list of folder types

The **FLDFolderTypeList** object provides a list of folder types authorized for logged-on user. For more information, see "Methodology for retrieving information" on page 13.

Constructor: FLDFolderTypeList(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Set/Get which records returned in the folder type list

The following methods set or get information on records are returned by the server.

setRequestAll

Sets the type of list returned by the server. The default is True.

Method: setRequestAll(boolean reqAll)

Parameter: "reqAll" set to:

- True, the server returns all folder types authorized for the logged-on user. This is the default.
- False, the server returns only those folder types that have folders that contain documents.

getRequestAll

Method: getRequestAll()

Returns: boolean. The setting executed with the setRequestAll method.

Navigate in the list

The following methods position at a record in the list. The navigation methods return boolean: True, if another record exists; False, if at the end of the list.

readFirst

Accesses the first record in the list.

Method: readFirst() **

readNext

Accesses the next record in the list. Assumes a readFirst or readPosition has been executed.

Method: readNext() **

readPosition

Accesses the record that contains the folder type specified by the folderType parameter, or if the folder type is not found, positions to the next folder type in the list.

Method: readPosition(int folderType) **

Parameter: "folderType" is the number of the folder type to be accessed.

Access a record

The following method populates an instance of the **FLDFolderTypeListEntry** interface with information contained in the record. It assumes a navigation method has been executed successfully.

getFolderTypeListEntry

Method: getFolderTypeListEntry()

Get the information in the record

When the getFolderTypeListEntry method is executed, the following methods return information contained in an instance of the **FLDFolderTypeListEntry** interface.

getFolderType

Method: getFolderType()

Returns: int. The folder type.

getTypeDescription

Method: getTypeDescription()

Returns: String. The folder type description.

Retrieve a list of document types

The **FLDDocumentTypeList** object provides a list of document types defined in the server database.

Constructor: FLDDocumentTypeList(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Navigate in the list

The following methods position at a record in the list. The navigation methods return boolean: True, if another record exists; False, if at the end of the list.

readFirst

Accesses the first record in the list.

Method: readFirst() **

readNext

Accesses the next record in the list. Assumes a readFirst or readPosition has been executed.

Method: readNext() **

readPosition

Accesses the record that contains the document type specified by the documentType parameter, or if the document type is not found, positions to the next document type in the list.

Method: readPosition(int documentType) **

Parameter: "documentType" is the document type to be accessed.

Access a record

The following method populates an instance of the **FLDDocumentTypeListEntry** interface with information contained in the record. It assumes a navigation method has been executed successfully.

getDocumentTypeListEntry

Method: getDocumentTypeListEntry()

Get the information in the record

When the getDocumentTypeListEntry method is executed, the following methods return information contained in an instance of the **FLDDocumentTypeListEntry** interface.

getDocumentType

Method: getDocumentType()

Returns: int. The document type.

getTypeDescription

Method: getTypeDescription()

Returns: String. The document type description.

Retrieve a list of mark types

The FLDMarkList object provides a list of mark types defined in the server database. This object is valid when connected to an RSD Folders server, version 4.2 or higher running on a Unix or Windows server platform where Mark functionality is implemented.

Constructor: FLDMarkList(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Methods

read

This method must be executed before applying other methods for this object.

Method: read()

getMarkVector

Retrieves a list of the marks types that the logged-on user is authorized to view.

Method: getMarkVector()

Returns: java.util.Vector. A list of mark types. Each vector element implements an instance of the FLDMarkListEntry interface. The methods in this interface are documented below.

getNumberOfMarks

Method: getNumberOfMarks()

Returns: int. The number of mark types defined on the server.

FLDMarkListEntry interface methods

getMarkType

Method: getMarkType()

Returns: short. The mark type number, a value from 1 to 16.

getMarkDescription

Method: getMarkDescription()

Returns: String. The description for the mark type.

getMarkLabel_0

Method: getMarkLabel_0()

Returns: String. The label when the mark status is set to 0.

getMarkLabel_1

Method: getMarkLabel_1()

Returns: String. The label when the mark status is set to 1.

isMarkActive

Method: isMarkActive(int documentType)

Parameter: "documentType" is the document type.

Returns: boolean. True if the mark is applied for the specified document type. False otherwise.

isSetMarkTo_1_Authorized

Method: isSetMarkTo_1_Authorized()

Returns: boolean. True if the logged-on user is authorized to set the mark status to 1. False otherwise.

isSetMarkTo_0_Authorized

Method: isSetMarkTo_0_Authorized()

Returns: boolean. True if the logged-on user is authorized to set the mark status to 0. False otherwise.

Create a folder

The **FLDCreateFolder** object creates a folder. The logged-on user must have authorization to create a folder. The isFolderCreateAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Constructor: FLDCreateFolder(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

createFolder

Method: createFolder(String folderName, int folderType) **

Parameters:

- "folderName" is the folder name, maximum 32 characters.
- "folderType" is a valid folder type defined on the server. For folder type information, see "Retrieve a list of folder types" on page 34.

Retrieve a list of filters

The **FLDFilterList** object provides a list of filters valid for the logged-on user.

Constructor: FLDFilterList(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Navigate in the list

The following methods position at a record in the list. The navigation methods return boolean: True, if another record exists; False, if at the end of the list.

readFirst

Accesses the first record in the list.

Method: readFirst() **

readNext

Accesses the next record in the list. Assumes a readFirst or readPosition has been executed.

Method: readNext() **

readPosition

Accesses the record that contains the filter name specified by the filterName parameter, or if the filter name is not found, positions to the next filter name in the list.

Method: readPosition(String filterName) **

Parameter: "filterName" is the name of the filter. The maximum length is 8 characters.

Access a record

The following method populates an instance of the **FLDFilterListEntry** interface with information contained in the record. It assumes a navigation method has been executed successfully.

getFilterListEntry

Method: getFilterListEntry()

Get the information in the record

When the getFilterListEntry method is executed, the following methods return information contained in an instance of the **FLDFilterListEntry** interface.

getFilterDescription

Method: getFilterDescription()

Returns: String. The filter description.

getFilterName

Method: getFilterName()

Returns: String. The filter name.

getLastUpdate

Method: getLastUpdate()

Returns: String. The filter modification date/time. Format: YYYYMMDDHHMM.

Set a filter

The FLDSetFilter object sets a filter as the active filter, returns the default filter definition and allows modification to the definition. The active filter restricts the folders and documents that can be retrieved. The filters available depend on the filters assigned to the logged-on user in the RSD Folders server database. If a folder list object is already open, any change to the active filter can be applied to a valid instance of the FLDFolderList object.

Constructor: FLDSetFilter(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Procedure for using filter methods

Procedure for using filter methods

1 Required. Execute the **applyFilter** method to set the active filter. This method creates a valid instance of this object. The methods for the object are then available and the instance can be referenced by the FLDFolderList object.

2 Optional. Modify the filter.

The modifications must be in compliance with the currently active filter. In other words, they can only further limit the active filter. Further, they should be in compliance with the logged-on user's authorizations. For more information, see "Methodology for retrieving information" on page 13.

For set and get methods related to modifications, see methods below.

3 Required to activate modifications. Execute **applySubFilter** to record the modifications to the active filter.

4 To apply the filter to an instance of the folder list object, execute the **setFolderList** method.

Grouping of methods initial

Note that the methods are grouped together, based on the information set and retrieved and not in strict alphabetic order.

- "Initialize and modify a filter" - set and modify a filter, apply current settings to a folder list.
- "Document and folder name criteria" on page 39 - set and retrieve document/folder name and matching criteria
- "Apply filter to folder criteria" on page 41 - set and retrieve ApplyFilterToFolder criteria.
- "Document category criteria" on page 42 - set and retrieve document category criteria.
- "Document/folder type and description criteria" on page 42 - set and retrieve folder/document type and descriiption criteria.
- "Mark status criteria" on page 43 - set and retrieve mark status criteria.
- "Date criteria" on page 43 - set and retrieve date criteria

Initialize and modify a filter

applyFilter

Sets the specified filter as the **active** filter and returns the default filter definition.

Method: applyFilter(String filterName) **

Parameter: "filterName" is a valid filter name, maximum length 8 characters. The filterName can be obtained using the FLDFilterList object. For more information, see "Retrieve a list of filters" on page 37.

applySubFilter

Modifies the active filter. Before using this method, execute methods to change the filter criteria.

Method: applySubFilter() **

setFolderList

If a folder list object is open and the active filter is either changed (to another filter name) or modified, this method applies the current filter to the instance of the specified folder list object. The object can then be queried again.

Method: setFolderList(FLDFolderList fldList) **

Parameter: "fldList" is a valid instance of the FLDFolderList object. For more information, see "Retrieve a list of folders" on page 47.

Document and folder name criteria

The following methods retrieve or set information related to the document name and folder name criteria.

Returning folders/documents that match or do not match the criteria

If connected to an RSD Folders server version 5.0 or higher running in a Unix or Windows server environment, the server can return a list of folders/documents that either match or that do not match the folder name/document name criteria.

The ability to change the folder/document list returned to either matching or not matching is only available if the folder/document name is not defined in the default filter definition. If a folder/document name is defined in the default criteria, then the default for 'NotMatching' cannot be changed.

IMPORTANT: If connected to a server running in a z/OS environment or an earlier version of the product in a Unix or Windows server environment, the list of folders/documents returned always matches the name criteria. Thus, in this section, the methods that have names that include 'NotMatching' are not applicable. For example, setDocumentNameNotMatching.

isNotMatchingFilterSupported

Method: isNotMatchingFilterSupported()

Returns: boolean.

- True, if the server can return a list of folders/documents that either match or that do not match the folder name/document name criteria
- False, if the server can only return a list of folders/documents that match the folder name/document name criteria

getDefaultDocumentName

Method: getDefaultDocumentName()

Returns: String. The default document name.

setDocumentName

Method: setDocumentName(String docName) **

Parameter: "docName" is the document name criteria.

getDocumentName

Method: getDocumentName()

Returns: String. The document name in the active filter.

isDefaultDocumentNameNotMatching

Method: isDefaultDocumentNameNotMatching()

Returns: boolean. True, the documents returned will be those with names not matching the document name criteria. False, the documents returned will be those with names matching the document name criteria.

isDocumentNameNotMatchingModifiable

Method: isDocumentNameNotMatchingModifiable()

Returns: boolean. True, the method setDocumentNameNotMatching can be executed. False otherwise.

setDocumentNameNotMatching

To use this method, the isDocumentNameNotMatchingModifiable method must return True.

Method: setDocumentNameNotMatching(boolean notMatching) **

Parameter: "notMatching" if set to True the documents returned will be those with names not matching the document name criteria. If set to False the documents returned will be those with names matching the document name criteria.

isDocumentNameNotMatching

Method: isDocumentNameNotMatching()

Returns: boolean. The value set using the setDocumentNameNotMatching method. True, the documents returned will be those with names not matching the document name criteria. False, the documents returned will be those with names matching the document name criteria.

getDefaultFolderName

Method: getDefaultFolderName()

Returns: String. The default folder name in the active filter.

setFolderName

Method: setFolderName(String folderName) **

Parameter: "folderName" is a valid folder name within the folder type.

getFolderName

Method: getFolderName()

Returns: String. The folder name in the active filter.

isDefaultFolderNameNotMatching

Method: isDefaultFolderNameNotMatching()

Returns: boolean. True, the folders returned will be those with names not matching the folder name criteria. False, the folders returned will be those with names matching the folder name criteria.

isFolderNameNotMatchingModifiable

Method: isFolderNameNotMatchingModifiable()

Returns: boolean. True, the method setFolderNameNotMatching can be executed. False otherwise.

setFolderNameNotMatching

To use this method, the isFolderNameNotMatchingModifiable method must return True.

Method: setFolderNameNotMatching(boolean notMatching) **

Parameter: "notMatching" if set to True the folders returned will be those with names not matching the folder name criteria. If set to False the folders returned will be those with names matching the folder name criteria.

isFolderNameNotMatching

Method: isFolderNameNotMatching()

Returns: boolean. The value set using the setFolderNameNotMatching method. True, the folders returned will be those with names not matching the folder name criteria. False, the folders returned will be those with names matching the folder name criteria.

Apply filter to folder criteria

getDefaultApplyFilterToFolder

Method: getDefaultApplyFilterToFolder()

Returns: boolean. True, displays only those folders that contain documents that meet the document criteria in the filter. False, displays all folders that meet the general filter definition.

setApplyFilterToFolder

The logged-on user must be authorized. to execute this action. See 'isApplyFilterToFolderModifyAuthorized' following.

Method: setApplyFilterToFolder(boolean applySubFilterToFolder) **

Parameter: "applySubFilterToFolder" set to:

- True, displays only those folders that contain documents that meet the filter criteria.
- False, displays all folders that meet the general filter definition.

isApplyFilterToFolder

Method: isApplyFilterToFolder()

Returns: boolean. See above

isApplyFilterToFolderModifyAuthorized

Method: isApplyFilterToFolderModifyAuthorized()

Returns: boolean. True, is the logged on user is authorized for this action. False, otherwise.

Document category criteria

getDefaultDocumentCategory

Method: getDefaultDocumentCategory()

Returns: byte[]. FLD_CATEGORY_ALL (includes all documents), FLD_CATEGORY_PC (includes only documents originating on a PC), FLD_CATEGORY_SERVER (includes only documents originating on a server).

setDocumentCategory

Method: setDocumentCategory(byte[] docCategory) **

Parameter: "docCategory" is one of the predefined variables below.

- FLD_CATEGORY_ALL (includes all documents)
- FLD_CATEGORY_PC (includes only document originating on a PC)
- FLD_CATEGORY_SERVER (includes only document originating on a server)

getDocumentCategory

Method: getDocumentCategory()

Returns: byte[]. See above.

Document/folder type and description criteria

getDefaultDocumentDescription

Method: getDefaultDocumentDescription()

Returns: String. The default document description for the filter.

getDefaultDocumentType

Method: getDefaultDocumentType()

Returns: int. The default document type.

setDocumentType

Method: setDocumentType(int docType) **

Parameter: "docType" is the document type criteria. For document type information, see "Retrieve a list of document types" on page 35.

getDocumentType

Method: getDocumentType()

Returns: int. The document type in the active filter.

getDefaultFolderDescription

Method: getDefaultFolderDescription()

Returns: String. The default folder description for default folder type.

getDefaultFolderType

Method: getDefaultFolderType()

Returns: int. The default folder type for the filter.

setFolderType

Method: setFolderType(int folderType) **

Parameter: "folderType" is a valid folder type. For folder type information, see "Retrieve a list of folder types" on page 34.

getFolderType

Method: getFolderType()

Returns: int. The folder type in the active filter.

Mark status criteria

setMarkStatus

This method can be executed several times to set more than one status. The logged-on user must be authorized to execute this method. The isFilterMarkModifyAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Method: setMarkStatus(short markType, int status) **

Parameters:

- "markType" is the mark identification number. Permissible values: 1 - 16.
- "status" is one of the following:
  - MARK_STATUS_MARKED_1 - includes only document where the specified mark has a status of 1.
  - MARK_STATUS_MARKED_0 - includes only document where the specified mark has a status of 0.
  - MARK_STATUS_APPLIED - includes only document where the specified mark is applied.
  - MARK_STATUS_NOT_APPLIED - includes only document where the specified mark is not applied.

isMarkSelectionActive

Method: isMarkSelectionActive(short markType)

Parameters: See setMarkStatus method above.

Returns: True, if filtering is set for the markType. False, otherwise.

isMarkStatus

Method: isMarkStatus(short markType, int status)

Parameters: See setMarkStatus method above.

Returns: True, if markType is set to the specified status in the filter. False, otherwise.

resetMarkSelection

Resets any selections made using the setMarkStatus method.

Method: resetMarkSelection()

Date criteria

The logged-on user must be authorized to modify dates in the filter. The isFilterDateModifyAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Setting the type of date

There are two types of dates associated with a document that can be used to limit a document list: the creation date and the working date. The document list can be filtered based on one of these dates, but not both at the same time.

The default is to use the creation date. Execute the method below to filter on the working date.

getDefaultUseWorkingDate

Method: getDefaultUseWorkingDate()

Returns: boolean. The default setting for the filter.

- True, indicates that the "working" date is used when applying date criteria.
- False, indicates that the "document creation" date is used when applying date criteria.

setUseWorkingDate

Method: setUseWorkingDate(boolean dateType) **

Parameter: "dateType" if True uses working date for filtering. If False, uses the creation (insertion) date for filtering. Default: False.

isUseWorkingdate

Method: isUseWorkingdate()

Returns: True, the date being filtered is the working date. False, the date being filtered is the creation date.

Setting date restrictions

For the type of date, the boundary can be set to restrict documents to those associated with dates that fall within a particular range. These are referred to as setting the To and From dates. You can set both of these dates or only one of them. For example, you can set only the To date with the result that all documents with dates after the date are shown.

There are two methods for setting the date:

- To simply indicate the value for a date, use the setFromDate and/or the setToDate method.
- Supported for RSD Folders server version 4.2 or higher, use the setFilterFromDate and/or the setFilterToDate to indicate a value or relative date.

getDefaultFilterFromDate

Method: getDefaultFilterFromDate()

Returns: java.util.Date. The default "from" date/time for the filter.

getDefaultFromDate

Method: getDefaultFromDate()

Returns: String. The default "from" date/time for the filter. Format: YYYYMMDDHHMM.

setFilterFromDate

Method: setFilterFromDate(FLDSetFilter.Date fromDate) **

Parameter: "fromDate" is the date set using the methods in the FLDSetFilter.Date interface (page 45).

getFilterFromDate

Method: getFilterFromDate()

Returns: The date set using the setFilterFromDate method.

getDefaultFilterToDate

Method: getDefaultFilterFromDate ()

Returns: java.util.Date. The default "to" date/time for the filter.

getDefaultToDate

Method: getDefaultToDate()

Returns: String. The default "to" date/time for the filter. Format: YYYYMMDDHHMM.

setFilterToDate

Method: setFilterToDate(FLDSetFilter.Date toDate) **

Parameter: "toDate" is the date set using the methods in the FLDSetFilter.Date interface (page 45).

getFilterToDate

Method: getFilterToDate()

Returns: The date set using the setFilterToDate method.

setFromDate

Method: setFromDate(String fromDate) **

Parameter: "fromDate" is the from date/time criteria. Format: YYYYMMDDHHMM.

getFromDate

Method: getFromDate()

Returns: String. The date set using the setFromDate method.

setToDate

Method: setToDate(String toDate) **

Parameter: "toDate" is the to date/time criteria. Format: YYYYMMDDHHMM.

getToDate

Method: getToDate()

Returns: String. The date set using the setToDate method.

FLDSetFilter.Date interface

The setFilterFromDate and setFilterToDate methods use, as input, values set in an instance of the FLDSetFilter.Date interface. The date can be set as absolute or relative.

Methods for both absolute and relative date

resetDate

Resets the date specifications set for either an absolute or relative date.

Method: resetDate()

setHours

Method: setHours(int hours)

Parameter: "hours" is the hours time for the date.

getHours

Method: getHours()

Returns: int. The hours time as set by the setHours method.

setMinutes

Method: setMinutes(int minutes)

Parameter: "minutes" is the minutes time for the date.

getMinutes

Method: getMinutes()

Returns: int. The minutes time as set by the setMinutes method.

setSeconds

Method: setSeconds(int minutes)

Parameter: "seconds" is the seconds time for the date.

getSeconds

Method: getSeconds()

Returns: int. The seconds time as set by the setSeconds method.

Absolute date format methods

For an absolute date, the date associated with the document must match the date specified using the setAbsoluteDate() method.

setAbsoluteDate

Limits documents to those that have a date that match the date associated with the document.

Method: setAbsoluteDate(java.util.Date specifiedDate)

Parameter: "specifiedDate" is the date.

getAbsoluteDate

Method: getAbsoluteDate()

Returns: java.util.Date. The date

Relative date format methods

For a relative date, the value is calculated based on the current computer date. For example if the relative type is days, the relative value is 3, and the current computer date is January 5, then the date value is calculated to be January 2.

To set any relative date:

1. Execute the setRelativeDate method.
2. Execute both the setRelativeType and the setRelativeValue methods.
3. Optionally, the setRelativeDateInTheFuture method can be executed.

setRelativeDate

Sets the date as relative or absolute.

Method: setRelativeDate(boolean relativeDate)

Parameter: "relativeDate" set to: True, the date is relative; False, the date is absolute. Default: False.

isRelativeDate

Method: isRelativeDate()

Returns: boolean. True, if the date is relative. False, if the date is absolute.

setRelativeType

Indicates the time period to use in conjunction with the setRelativeValue.

Method: setRelativeType(int relativeType)

Parameter: "relativeType" is one of the following constants: RELATIVE_VALUE_TYPE_DAYS (default), RELATIVE_VALUE_TYPE_WEEKS, RELATIVE_VALUE_TYPE_MONTHS.

getRelativeType

Method: getRelativeType()

Returns: int. The time period set.

setRelativeValue

Indicates the value to use in conjunction with the time period.

Method: setRelativeValue(int relativeValue)

Parameter: "relativeValue" is the number of days, weeks or months. Default: 0.

getRelativeValue

Method: getRelativeValue()

Returns: int. The number used in conjunction with the time period.

setRelativeDateInTheFuture

Indicates the whether the relative value is positive (in the future) or negative (in the past). Valid only for a working date.

Method: setRelativeDateInTheFuture(boolean futureValue)

Parameter: " futureValue" set to: True, the calculated date is after the current computer date; False, the calculated date is before the current computer date. Default: False.

isRelativeDateInTheFuture

Method: isRelativeDateInTheFuture()

Returns: boolean. True, the relative date is in the future. False, the relative value is in the past.

Retrieve a list of folders

The **FLDFolderList** object returns a list of folder names that meet the active filter criteria and the logged-on user's authorizations. For more information on the default and valid folder types, see "Methodology for retrieving information" on page 13.

Constructor: FLDFolderList(FLDSetFilter setFilter) **

Constructor: FLDFolderList(FLDEnvironment environment, FLDSetFilter setFilter) **

Parameters:

- "setFilter" is a valid FLDSetFilter object. For more information, see "Set a filter" on page 38.
- "environment" is a valid instance of the FLDEnvironment interface. Useful only if connected to a Global View Monitor. For more information, see "List and set environments" on page 23.

Include or suppress associated folders

Optionally, associated folders can be included in the list. Associated folders are listed directly under the principle folder name in the list.

setShowAssociatedFolders

Forces or suppresses the display of associated folders in the list. If associated folders are displayed, the principle folder is listed first with the associated folders listed directly below it.

Method: setShowAssociatedFolders(boolean showAssociatedFolders)

Parameter: "showAssociatedFolders" set to:

- True, to include associated folders in the list.
- False, to suppress associated folders. This is the default.

getShowAssociatedFolders

Returns the status for showing associated folders.

Method: getShowAssociatedFolders(boolean showAssociatedFolders)

Returns: Boolean. See setShowAssociatedFolders (above) for values.

Navigate in the list

The following methods position at a record. The list is in ascending alphabetic order. The navigation methods return boolean: True, if another record exists; False, if at the beginning or end of the list.

Parameters for all navigation methods:

- "fldType" must fall within the range of folder types defined in the active filter and valid for logged-on user. For more information, see "Set a filter" on page 38.
- "fldName" is a folder name.
- "assdFldType" is the folder type for the associated folder. If not displaying associated folders, set this parameter to zero.
- "assdFldName" is the folder name for the associated folder. If not displaying associated folders, set this parameter to "".

readFirst

The methods listed below access the first record in the list. The first method uses the default folder type and the second the folder type specified in the parameter.

Method: readFirst() ** or readFirst(int fldType) **

readLast

The methods listed below access the last record in the list. The first method uses the default folder type and the second the folder type specified in the parameter.

Method: readLast() ** or readLast(int fldType) **

readNext

The methods listed below access the next record in the list for the folder type specified by the readFirst or readPosition method previously executed.

Method: readNext() ** or readNext(String fldName, int assdFldType, String assdFldName) **

readPosition

The methods listed below access the record in the list that matches the parameters, or if the folder is not found, positions to the next folder in the list. The first method uses the default folder type and the second the folder type specified in the parameter.

Method: readPosition(String fldName) **

Method: readPosition(int fldType, String fldName) **

readPrevious

The methods listed below access the previous record in the list for the folder type specified by the readLast or readPosition method previously executed.

Method: readPrevious() ** or readPrevious(String fldName, int assdFldType, String assdFldName) **

Access a record

getFolderListEntry

Method: getFolderListEntry()

Returns: An instance of the **FLDFolderListEntry** interface with information contained in the record.

Upload a document

The newUploadTransfer method implements and Instance of the FLDUploadTransfer interface, facilitating the uploading of the specified file. For more information, see "Upload a file" on page 53.

newUploadTransfer

Method: newUploadTransfer(FLDFolderListEntry folderListEntry, java.lang.String documentName, int documentType, java.lang.String fileNameExt, java.util.Date workingDate, java.lang.String descriptorName)

Method: newUploadTransfer(java.lang.String folderName, int folderType, java.lang.String documentName, int documentType, java.lang.String fileNameExt, java.util.Date workingDate, java.lang.String descriptorName)

Parameters:

- "folderListEntry" is a valid instance of the FLDFolderList object.
- "folderName" is a valid folder name within the folder type.

  "documentName" is the name to be assigned to the document in when it is inserted in the database.
- "documentType" the number corresponding to a document type defined in the database. For a list of document types, see "Retrieve a list of document types" on page 35.
- "fileNameExt" is the extension for the document.
- "workingDate" is the working date to assign to the document.
- "descriptorName" is the name of a valid descriptor. For a list of descriptors, see "Retrieve a descriptor list" on page 52.

Get the information in the record

When the getFolderListEntry method is executed, the following methods return information contained in an instance of the **FLDFolderListEntry** interface.

In addition to retrieving information, this interface contains a method to rename a folder.

getAssociatedToFolderName

Method: getAssociatedToFolderName()

Returns:

- String. The folder name or
- null. If the folder is not an associated folder.

getAssociatedToFolderType

Method: getAssociatedToFolderType()

Returns:

- int. The folder type or
- null. If the folder is not an associated folder.

getFolderDate

Method: getFolderDate()

Returns: java.util.Date. The date the folder was created.

createFolderExtract

Implements an instance of the FLDExtract interface, which permits the extraction of the folder. For more information, see "Extract a folder or document" on page 120. If the logged-on user is not authorized for this action, returns null.

Method: createFolderExtract()

Note: To execute this method, the isExtractAuthorized method must return True.

getFolderName

Method: getFolderName()

Returns: String. The folder name.

getFolderStatus

Method: getFolderStatus()

Returns: int. The folder status.

Possible values:

- FLD_ASSOC_WITH_ASSOC (indicates an associated folder with associated folders)
- FLD_ASSOCIATED (indicates an associated folder)
- FLD_INTERNAL_PROC (used for internal program processing)
- FLD_STANDARD (indicates a standard folder)
- FLD_STD_WITH_ASSOC (indicates a standard folder with associated folders.)

getFolderType

Method: getFolderType()

Returns: int. The folder type.

getNumberDocuments

Method: getNumberDocuments()

Returns: int. The number of documents in folder.

getNumberPages

Method: getNumberPages()

Returns: int. The number of pages in folder.

isAssociated

Method: isAssociated()

Returns: boolean. True, if the folder is associated to another folder. False, otherwise.

rename

The logged-on user must be authorized for this functionality.

Method: rename(String folderName) **

Parameter: "folderName" is the new name to assign to the folder.

isExtractAuthorized

Method: isExtractAuthorized()

Returns: boolean. True, the logged on user is authorized for this action. False, otherwise.

isRenameAuthorized

Method: isRenameAuthorized()

Returns: boolean. True, if the logged-on user is authorized to rename a folder. False, otherwise.

Document objects

Section overview

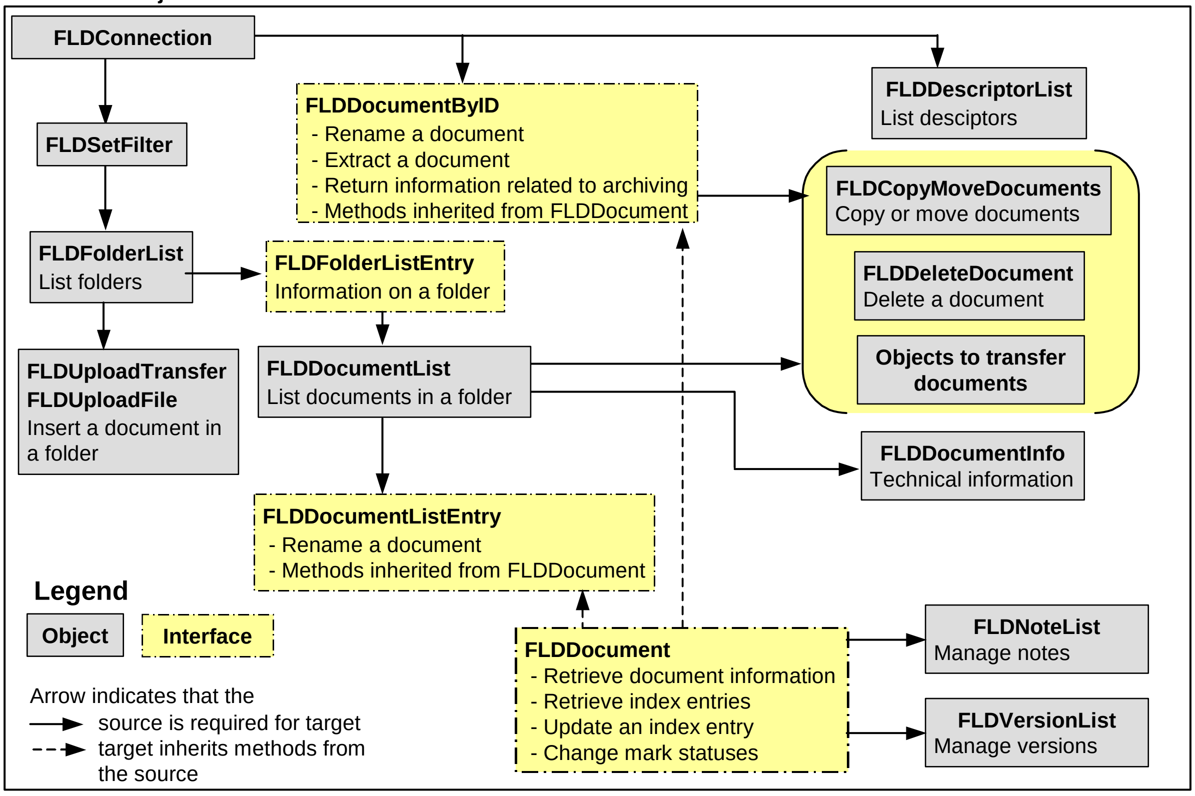
Below is a list of the objects used to retrieve a list of descriptors, upload a file, retrieve a list of documents, retrieve information on a single document and delete a document.

Objects covered in this section

- "Retrieve a descriptor list" on page 52. FLDDescriptorList supplies a list of descriptors, useful when uploading a file.
- "Upload a file" on page 53. FLDUploadFile uploads a file to the server.
- "Retrieve a list of documents" on page 56. FLDDocumentList retrieves a list of documents.
- "Retrieve document information" on page 60. Explains the different interfaces used to retrieve information on a document: FLDDocumentListEntry, FLDDocument and FLDDocumentInfo, FLDDocumentByID.
- "Copy or move a document" on page 69. FLDCopyMoveDocuments copies or moves one or more documents to another folder.
- "Delete a document" on page 70. FLDDeleteDocument deletes a document.
- "Manage notes" on page 71. Objects and interfaces to create, access, modify and delete notes that are associated with a document.
- "Manage document versions" on page 76. Objects and interfaces to create, access and delete document versions and to manage version labels..

The image below is designed to show the objects related to FLDDocumentList and FLDDocument. Note that some related functionality, for example deleting or transferring a document, is also available from a query list.

Document objects



Retrieve a descriptor list

The **FLDDescriptorList** object provides a list of descriptors defined in the server database. This information is useful to upload a document to the server (the FLDUpload object). For more information, see "Upload a file" on page 53.

- For RSD Folders Server running on a z/OS platform, these are the input descriptors, which in turn point to the archive descriptor associated with it.
- For RSD Folders Server running on a Unix or Windows server platform, these are the archive descriptors.

Constructor: FLDDescriptorList(FLDConnection connectObject)

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Access a record in the list

The following methods access a record in the list and populate the object with information contained in the record. The methods return boolean: True, if another record exists; False, if at the end of the list.

ReadFirst

Accesses the first record in the list.

Method: readFirst() **

readNext

Accesses the next record in the list. Assumes a readFirst has been executed.

Method: readNext() **

Get the information in the record

When the method readFirst or readNext is performed, the following methods return information contained in that record in the list.

getDescriptorName

Method: getDescriptorName()

Returns: String. The descriptor name.

getDescriptorComment

Method: getDescriptorComment()

Returns: String. The descriptor description/comment.

Upload a file

There are 2 objects available for uploading a document to the server: FLDUploadTransfer and FLDUploadFile.

- FLDUploadTransfer manages the upload process and uploads the complete file.
- FLDUploadFile requires the programmer to manage the process.

For both:

- The logged-on user must have authorization to create a document. The isDocCreateAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.
- If the folder name does not exist, it will be created.
- If the document is in PDF format, it must not be secured using Adobe or compatible security.
- Restriction for the file extension:
  - Maximum length is 3 characters.
  - If the extension is exactly 3 characters, it cannot contain any numbers.

FLDUploadTransfer

When the NewUploadTransfer method is executed (FLDFolderLIst), an instance of the **FLDUploadTransfer** interface is implemented. For more information on this method, see "Upload a documen" on page 49.

The following methods are for internal use: setSAPAccessRight, getSAPAccessRight, setInputDescriptor, getInputDescriptor.

Set methods

The first time that an instance of the FLDUploadTransfer interface is used to upload a document, the set methods do not need to be executed because the information is already defined in the NewUploadTransfer method. However, the set methods must be executed if the same instance is used to upload additional documents.

setDescriptor

Method: setDescriptor(String descriptortName)

Parameter: "descriptorName" is the name of a valid descriptor. For a list of descriptors, see "Retrieve a descriptor list" on page 52.

setDocumentName

Method: setDocumentName(String documentName)

Parameter: "documentName" is the name to be assigned to the document when it is inserted in the database.

setDocumentType

Method: setDocumentName(int documentType)

Parameter: "documentType" is the number corresponding to a document type defined in the database. For a list of document types, see "Retrieve a list of document types" on page 35.

setFileNameExtension

Method: setFileNameExtension(String fileExtension)

Parameter: "fileExtension" is the extension for the document.

setWorkingDate

Method: setWorkingDate(java.util.Date workingDate)

Parameter: "workingDate" is the working date assigned to the document.

Transfer methods

transfer

Method: transfer(java.io.File file)

Parameter: "file" is the full path and filename of the file to upload.

Method: transfer(java.io.I nputStream inputStream)

Parameter: "inputStream" is the buffer with the information to upload.

Get methods

getDescriptor

Method: getDescriptor

Returns: String. The descriptor name.

getDocumentID

Method: getTextDocumentID

Returns: String. The document identifier assigned by the server.

getDocumentKey

Method: getDocumentKey

Returns: Byte[]. The document key assigned by the server.

getDocumentName

Method: getDocumentName

Returns: String. The document name.

getDocumentType

Method: getDocumentName

Returns: int. The document type.

getExtendedDocumentKey

Method: getExtendedDocumentKey

Returns: int. The extended document key assigned by the server.

getFileNameExt

Method: getFileNameExt

Returns: String. The document extension.

getOutputStream

Method: getOutputStream

Returns: java.io.OutputStream. The output stream.

getRawDocumentID

Method: getDocumentID

Returns: Byte[]. The document identifier assigned by the server.

getWorkingDate

Method: getWorkingDate(workingDate)

Returns: java.util.Date. The working date.

FLDUpload

There are two constructors.

Constructor: FLDUploadFile(FLDFolderList folderList) **

Constructor: FLDUploadFile(FLDFolderList folderList, String folderName, int folderType, String document name) **

Parameters:

- "folderList" is a valid instance of the FLDFolderList object.
- "folderName" is a valid folder name within the folder type.
- "folderType" is a valid folder type.

  "documentName" is the name to be assigned to the document in when it is inserted in the database.

Upload a document

Before executing any write method, the developer must manually transfer the contents of the file to a buffer. Then the writeFirst method must be executed to create a document entry and the writeNext method must be executed as many times as necessary to transfer the contents of the buffer.

The methods return: boolean: True, if the write is successful; False, otherwise.

writeFirst

Creates a document entry in the database.

Method: writeFirst(String folderName, int folderType, String documentName, int docType, String pcFileName, String workingDate, String descriptor) **

Parameters:

- For "folderName", "folderType" and "documentName", see parameters for the constructor.
- "docType" the number corresponding to a document type defined in the database. For a list of document types, see "Retrieve a list of document types" on page 35.
- "pcFileName" is the path and name of the PC file to be uploaded, maximum 128 characters.
- "workingDate" is the working date to assign to the document. Format: YYYYMMDD.
- "descriptor" is the name of a valid descriptor. For a list of descriptors, see "Retrieve a descriptor list" on page 52.

writeNext

Transfers the contents of a buffer to the server.

Method: writeNext(int bufferSize, byte[] buffer, boolean isLastBlock) **

Parameters:

- "bufferSize" is the size of the buffer. Maximum size is 32,000 bytes.
- "buffer" indicates the array in bytes that contains the data to upload.
- "isLastBlock". True if this is the last block of data to be transferred. False, otherwise.

Get information on the upload

The following methods can be used after the process of uploading the document has been successfully completed.

getDocumentID

Method: getDocumentID()

Returns: String. Length, 128 characters. If document ID functionality is activated on the server, then a unique document id assigned to the document that is created as a result of the upload. For more information on document id functionality, see "Unique document ID functionality" on page 20.

getDocumentKey

Method: getDocumentKey()

Returns: byte[]. The document key assigned to the document that is created as a result of the upload.

getExtendedDocumentKey

Method: getExtendedDocumentKey()

Returns: short. The extended document key. Permissible values: the extended document key (when accessing a server defined as a Global View Monitor), otherwise 0.

getRawDocumentID

Method: getRawDocumentID()

Returns: byte[]. Length, 64 bytes. If document ID functionality is activated on the server, then a unique document id assigned to the document that is created as a result of the upload. For more information on document id functionality, see "Unique document ID functionality" on page 20.

getRestartOffset

Method: getRestartOffset()

Returns: int. The size of the data successfully received by the server.

Retrieve a list of documents

The **FLDDocumentList** object provides a list of document names for the specified folder type and folder name. It returns information on the number of documents in the list and on each document.

Constructor: FLDDocumentList(FLDFolderList folderList) **

Parameter: "folderList" is a valid instance of the FLDFolderList object.

Define information to return

setRunExtensionInfo

This method provides the possibility to retrieve information on the run date and run number in an instance of the FLDDocumentListEntry interface. It must be executed before using any navigation method. For more information, see "getRunDate" on page 63. Note that this information can always be retrieved in an instance of the FLDDocumentInfo object.

Method: setRunExtensionInfo(boolean runInfo)

Parameter: "runInfo" if True, the additional information is retrievable in an instance of the FLDDocument interface. If false, this information cannot be retrieved from this interface.

setDocumentIDExtensionInfo

Method: setDocumentIDExtensionInfo (boolean docInfo)

Parameter: "docInfo" set to:

- True, permits retrieval of the documentID and the rawDocument ID using methods in the FLDDocument interface.
- False, the information is not available.

Define list sort order

If supported on the server, these methods can be used to control the sort order for documents.

isSortable

This method can be used to verify that the functionality is supported on the server.

Method: isSortable()

Returns: boolean. True if the document list can be sorted. False, otherwise.

applySort

Sorts the document list according the criteria set using the FLDSort object or resets the list to the original sort order (by creation date). For more information, see "Sort objects" on page 57.

Method: applySort(FLDSort listSort) **

Parameter: "listSort" is an instance of the FLDSort object, where the sort criteria has been set. Note, if the paremeter is null, then the list is sorted in the default sort order.

resetSort

This method returns the document list sorted in the default sort order, by document creation date in ascending order.

Method: resetSort() **

Sort objects

There are two objects: use FLDSortCriteria to define the sort criteria and FLDSort to implement the use of this criteria. Sorting can only be implemented when accessing an RSD Folders server version:

- 4.3 in a z/OS enviroment.
- 4.2 patch 1.3 in a Unix or Windows server environment.

Set sort criteria

The **FLDSortCriteria** object defines the sort criteria that can be used to sort a document list. The constructors and methods available are listed below.

Constructor: FLDSortCriteria()

Constructor: FLDSortCriteria(int field, boolean ascending)

Parameter: "int" see setField method following. "ascending" see setSortAscending method following.

setField

This method sets the field to be sorted

Method: setField(int field)

Parameter: "field" is the field to sort.

For a document list, this is one of the following constants: DOCUMENT_NAME, DOCUMENT_SIZE, DOCUMENT_TYPE, PAGE_NUMBER, RUN_EXTENSION, WORKING_DATE, MARK_STATUS_X (where X is the number of the mark, 1 - 16).

setSortAscending

This methods sets the direction of the sort. Default: True.

Method: setSortAscending(boolean ascending)

Parameter: "ascending" set to True, the sort is in ascending order; False, sort is in descending order.

Default: True.

getField

Method: getField()

Returns: int. The constant set in the constructor or using the setField method.

isAscending

Method: isAscending()

Returns: boolean. True, the sort is ascending. False, the sort is descending.

Set sort

The **FLDSort** object implements the use of sort criteria. The constructor and methods are documented below.

Constructor: FLDSort()

Once an instance of this object has been initialized, the "applySort" method documented on page 56 method can be used to implement the sort of a list.

addSortCriteria

Each time this method is executed, another set of criteria is added. If more that one set of criteria is added, when the sort is implemented, the first set of criteria is the primary sort, the second set is the secondary sort, etc.

Method: addSortCriteria(FLDSortCriteria criteria)

Parameter: "criteria" is an instance of the FLDSortCriteria object.

removeAll

Resets all sort criteria

Method: removeAll()

Navigate in the list

The following methods position at a record in the list. The document list is ordered in ascending order by the date of insertion in the database. The navigation methods return boolean: True, if another record exists; False, if at the beginning or end of the list.

Note that the following methods are for internal use: isAtBottom, isAtTop.

Parameters for all navigation methods:

- "folderType" must fall within the range of folder types defined in the active filter and valid for the logged-on user. For folder type information, see "Retrieve a list of folder types" on page 34 and "Set a filter" on page 38.
- "folderName" is a valid folder name within the folder type.
- "documentKey" is the document key.
- "extendedDocumentKey" is the extended document key.
- "documentName" is the document name.
- "backwards" if set to True, searches towards the top of the list; if false searches towards the bottom of the list.

Note: For documentKey, extendedDocumentKey and docName, valid values can be returned using methods described in the topic "General document information" on page 60.

readFirst

Accesses the first record in the list.

Method: readFirst(String folderName, int folderType) **

readLast

Accesses the last record in the list.

Method: readLast(String folderName, int folderType) **

readNext

The methods listed below access the next record in the list for the folder type and folder name specified by the readFirst or readPosition method previously executed. The first method accesses the next record in the list and the second method accesses the record following the one that matches the specifications defined in the parameters.

Method: readNext() **

Method: readNext(byte[] documentKey, short extendedDocumentKey) **

Method: readNext(byte[] documentKey, short extendedDocumentKey, String documentName) **

readPrevious

The methods listed below access the previous record in the list for the folder type and folder name specified by the readFirst or readPosition method previously executed. The first method accesses the previous record and the second accesses the record previous to the one matches the specifications defined with the parameters.

Method: readPrevious() **

Method: readPrevious(byte[] documentKey, short extendedDocumentKey) **

Method: readPrevious(byte[] documentKey, short extendedDocumentKey, String documentName) **

readPosition

The first method searches for the document name starting at the beginning of the list and positions at the first occurrence of the name specified or, if not found, to the next document in the list.

The second method searches for the document name starting at the document specified by the documentKey parameter and searches in the direction specified by the backwards parameter. It then positions at the first occurrence of the name specified or, if not found, to the next document in the list.

Method: readPosition(String folderName, int folderType, String documentName) **

Method: readPosition(String folderName, int folderType, String documentName, byte[] documentKey, short extendedDocumentKey, boolean backwards) **

Get information on the document list

Once a "read" method has been executed, the following methods return information on the document list.

getFolderName

Method: getFolderName()

Returns: String. The name of the folder in which the documents are stored.

getFolderType

Method: getFolderType()

Returns: int. The folder type.

getNumberDocuments

Method: getNumberDocuments()

Returns: int. The number of documents retrieved.

Access a document record

Once a 'read' method has been executed, the getDocumentListEntry method must be used to retrieve information on the document.

getDocumentListEntry

Method: getDocumentListEntry()

Result: An instance of the FLDDocumentListEntry interface is populated with information contained in the record. For more information, see "Retrieve document information" on page 60.

Retrieve document information

From either a document list of a query list, you can retrieve document information.

- From a document list, the getDocumentListEntry method must be executed. This method creates an instance of the FLDDocumentListEntry interface. For information on the FLDDocumentListEntry interface, see below.
- From a query list, the next method must be executed. This method creates an instance of the FLDQueryListEntry interface. For information on the FLDQueryListEntry interface, see "Access a record" on page 101.

Document information can also be retrieved directly after a connection is made to the server using the getFLDDocumentByID method. This method implements an instance of the FLDDocumentByID interface. For more information, see "getFLDDocumentByID" on page 25 and "Direct access to document information" on page 67.

FLDDocumentListEntry interface

An instance of the interface returns document information when the getDocumentListEntry method is executed. For more information, see "getDocumentListEntry" on page 59.

For methods available once there is a valid instance of this interface, see

- "General document information" below
- "Technical document information" on page 65
- Additional methods available for this interface are documented following.

Note that the following methods are for internal use: createDynamicExtract, getSapData, isPCRecognizedExtension, isSapExtension.

createDocumentExtract

Implements an instance of the FLDExtract interface, which permits the extraction of the document. For more information, see "Extract a folder or document" on page 120. If the logged-on user is not authorized for this action, returns null. To execute this method, the isExtractAuthorized method must return True.

Method: createDocumentExtract()

isExtractAuthorized

Method: isExtractAuthorized()

Returns: boolean. True, the logged on user is authorized for this action. False, otherwise.

rename

The logged-on user must be authorized for this functionality. The isDocRenameAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Method: rename(String documentName) **

Parameter: "documentName" is the new name to assign to the document.

General document information

The **FLDDocument** interface contains methods that are inherited by instances of the FLDDocumentListEntry interface, the FLDQueryListEntry interface and the FLDDocumentByID interface. An instance of this object is referenced by a number of methods.

In addition to the methods documented in this topic, an instance of the FLDDocument interface facilitates access to managing notes associated with a document. For more information, see "Manage notes" on page 71.

Restrictions using the methods with the FLDQueryListEntry interface

The following methods only return information if, when the query is executed in FLDQueryList, the method "enableDocumentDetails", documented on page 98, is executed, set to true. Methods affected: getDocumentNumberPages, getDocumentSize, getDocumentType, getDocumentTypeDescription, getDocumentWorkingDate.

The following methods only return information if, when the query is executed in FLDQueryList, the method "enableRunInfo", documented on page 99, is executed, set to True. Methods affected: getRunDate, getRunNumber. Use the isRunInfoAvailable method (page 65) to verify that the run date and run number can be retrieved from this object.

The following methods only return information if, when the query is executed in FLDQueryList, the method "enableDocumentID", documented on page 99, is executed, set to True. Methods affected: getDocID, getRawDocID.

Restrictions on using the methods with the FLDDocumentListEntry interface

The following methods only return information if, in the instance of the FLDDocumentList object, the method "setRunExtensionInfo", documented on page 56, is executed, set to True. Methods affected: getRunDate, getRunNumber.

The following methods only return information if, in the instance of the FLDDocumentList object, the method "setDocumentIDExtensionInfo", documented on page 56, is executed, set to True. Methods affected: getDocID, getRawDocID.

- Use the isRunInfoAvailable method (page 65) to verify that the run date and run number can be retrieved from this object.
- From an instance of the FLDDocumentLIstEntry interface, the run number and run date can also be returned using methods in the FLDDocumentInfo object. For more information, see "Technical document information" on page 65.

Methods

Most methods return information. The exceptions are listed below.

- The method newIndexFieldUpdate (page 65) enables the addition, deletion or modification of an index entry value.
- The methods, changeMark and changeMarks, change the status of marks.
  - The methods are only valid for RSD Folders version 4.2 or higher running on a Unix or Windows server platform where Mark functionality is implemented.
  - The logged-on user must be authorized to execute the method. To verify authorizations, see "Retrieve a list of mark types" on page 36.
  - The mark types must be defined for that document type. To verify whether the mark type is valid for the current document:
    - Retrieve the document type using the documentType method (page 63).
    - Retrieve a list of mark types defined on the server and verify that the mark type is applied for the document type. For more information, see "Retrieve a list of mark types" on page 36.
  - Use the method getMaxNumberOfMarks (page 63) to verify the number of mark types applied to the document.

changeMark

The following changes the mark status for the mark type specified.

Method: changeMark(FLDMarkListEntry markType) **

Parameter: "markType" is an instance of FLDMarkListEntry.

changeMarks

The following changes the mark statuses for the mark types listed in the vector.

Method: changeMarks(java.util.Vector markList) **

Parameter: "markList" is a vector of instances of FLDMarkListEntry.

getDefaultExtension

Method: getDefaultExtension()

Returns: String. The default document extension, as text.

getDocID

Method: getDocID()

Returns: String. Length, 128 characters. For more information on document id functionality, see "Unique document ID functionality" on page 20.

getDocumentExtension

Method: getDocumentExtension()

Returns: String. The document extension that indicates the format of the document.

Returned values: (* indicates information for position 1 of the record)

Value	Indicate	Value	Indicates
000	Text: no carriage return *	210	Same as '200', but with custom resources
010	Same as '000', but with custom resources	300	AFP
100	Text: carriage return (ASA) *	400	Microfiche
110	Same as '100', but with custom resources	500	Text: not converted to EBCDIC
200	Text: carriage return (MACHINE) *	xxx	Binary: uses the actual PC extension (i.e. doc, xls, wks, pdf)

getDocumentIndexes

This method retrieves a list of indexes that are associated with a document.

Notes:

- The method is valid only when connected to an RSD Folders server running on a Unix or Windows server platform where ABI functionality is implemented.
- The logged user name must be authorized to retrieve index values. The method isIndexValueViewAuthorized must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.
- Use the hasIndexValues method to find out whether the document has associated index entries.

The first method returns a list containing all indexes that have entries associated with the document. The second method restricts the list to those defined by the parameter "indexes".

Method: getDocumentIndexes()

Method: getDocumentIndexes(java.util.List indexes)

Parameter: "indexes" is a list of index names.

Returns: java.util.List. The methods to return information on an entry in the list are contained in the FLDIndexValues interface. For more information, see "Get information on the indexes" on page 102.

getDocumentKey

Method: getDocumentKey()

Returns: byte[]. The server document key. Used internally to access the logical document.

getDocumentName

Method: getDocumentName()

Returns: String. The document name.

getDocumentNumberPages

Method: getDocumentNumberPages()

Returns: int. The number of pages in the document.

getDocumentSize

Method: getDocumentSize()

Returns: int. The size of the document in kilobytes.

getDocumentType

Method: getDocumentType()

Returns: int. The document type.

getDocumentTypeDescription

Method: getDocumentTypeDescription()

Returns: String. The description for the document type.

getDocumentWorkingDate

Method: getDocumentWorkingDate()

Returns: java.util.Date. The working date assigned to the document.

getExtendedDocumentKey

Method: getExtendedDocumentKey()

Returns: short. The extended document key.

- The value 0, if the server is not a Global View Monitor.
- A value, when accessing a server that is defined as a Global View Monitor.

getFolderName

Method: getFolderName()

Returns: String. The folder name.

getFolderType

Method: getFolderType()

Returns: int. The folder type.

getMaxNumberOfMarks

This method is only valid for RSD Folders version 4.2 or higher running on a Unix or Windows server platform where Mark functionality is implemented.

Method: getMaxNumberOfMarks()

Returns: int. The number of marks defined for the document.

getRawDocID

Method: getRawDocID()

Returns: byte[].Length, 64 bytes. For more information on document id functionality, see "Unique document ID functionality" on page 20.

getRunDate

Method: getRunDate()

Returns: java.util.Date. The run date and time. Use the isRunInfoAvailable method (page 65) to verify that the run date and run number can be retrieved from this object.

getRunNumber

Method: getRunNumber()

Returns: int. The run number. Use the isRunInfoAvailable method (page 65) to verify that the run date and run number can be retrieved from this object.

hasIndexValues

Method: hasIndexValues()

Returns: boolean. True if the document has index entries associated with it. False otherwise.

isAFP

Method: isAFP()

Returns: boolean. True if the document is in AFP format. False otherwise.

isASA

Method: isASA()

Returns: boolean. True if the document is a text document with a control character (ASA) in position 1 of the record. False otherwise.

isLocked

Method: isLocked()

Returns: boolean. True if the document is locked. False otherwise.

isMachine

Method: isMachine()

Returns: boolean. True if the document is a text document with a control character (ASA) in position 1 of the record. False otherwise.

isMail

Method: isMail()

Returns: boolean. True if the document is an e-mail. False otherwise.

isMailTNEF

For internal use.

isMailWithAttachment

Method: isMailWIthAttachment()

Returns: boolean. True if the document is an e-mail with an attachment. False otherwise.

isMarkStatus_1

Valid for RSD Folders version 4.2 or higher running on a Unix or Windows server platform where Mark functionality is implemented.

The method returns results on the status of the mark.

Method: isMarkStatus_1(short markType)

Returns: boolean. True, if the mark status is set to 1 for the document. False, if the mark status is set to 0 for the document or the mark type is not applied to the document.

Note: To ensure that the information returned is providing information on the actual mark status, it is highly recommended that the programmer verify that the mark type is applied to the document type associated with the current document. To make this verification, execute the method isMarkActive (FLDMarkListEntry interface). For more information, see "Retrieve a list of mark types" on page 36.

isMicrofiche

Method: isMicrofiche()

Returns: boolean. True if the document is in microfiche format. False otherwise.

isPC

Method: isPC()

Returns: boolean. True if the document is in binary format. False otherwise.

isPDF

Method: isPDF()

Returns: boolean. True if the document is in PDF format. False otherwise.

isRunInfoAvailable

Method: isRunInfoAvailable(()

Returns: boolean. True if information on the run date and run number is available. False, otherwise.

isText

Method: isText()

Returns: boolean. True if the document is in text format. False otherwise.

isTextAscii

Method: isTextAscii()

Returns: boolean. True if the document is in text format. False otherwise.

hasVersions

Method: hasVersions()

Returns: boolean. True if there is more than one version of the document. False otherwise.

newIndexFieldUpdate

There are two implementations of this method:

The method valid for RSD Folders version 4.2 or higher running on a Unix or Windows server platform where ABI functionality is implemented:

Method: newIndexFieldUpdate() **

Result: Implements an instance of the FLDIndexFieldUpdate interface. For more information, see "Add, delete or change index entries" on page 103.

The method valid for RSD Folders version 5.1 or higher running on a Unix or Windows server platform where ABI functionality is implemented:

Method: newIndexFieldUpdate(FLDIndexUpdateTransaction updateTransaction) **

Parameter: "updateTransaction" is a valid instance of the "FLDIndexUpdateTransation" object.

Result: Implements an instance of the FLDIndexFieldUpdate interface. For more information, see "Add, delete or change index entries" on page 103.

Technical document information

An instance of the FLDDocumentInfo object returns technical information on a document.

Constructor: FLDDocumentInfo(FLDDocumentList documentList) **

Parameter: "documentList" is a valid instance of the FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Access a record

read

Method: read(FLDDocument document) **

Parameter: "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.

Get the information in the record

When a read method is performed, the following methods return information for the specified document.

getArchiveExpDate

Method: getArchiveExpDate()

Returns: java.util.Date. The archive expiration date.

getArchiveExpirationDate

Method: getArchiveExpirationDate()

Returns: String. The archive expiration date. Format: YYYYMMDD.

getDescriptor

Method: getDescriptor()

Returns: String. The name of the descriptor used when the document is captured. For more information on the descriptor, see "Retrieve a descriptor list" on page 52.

getDocExtension

Method: getDocExtension()

Returns: String. The document extension that indicates the format of the document. For more information, see the method "getDocumentExtension" on page 62.

getDocID

Method: getDocID()

Returns: String. Length, 128 characters. The document id assigned to the document when it is created. This information is required when using the getFLDDocumentByID method.For more information, see "getFLDDocumentByID" on page 25.

getFicheNumber

Method: getFicheNumber()

Returns: String. The fiche number. Applies only when connected to RSD Folders Server running on a z/OS platform.

getFicheRunNumber

Method: getFicheRunNumber()

Returns: String. The fiche run number. Applies only when connected to RSD Folders Server running on a z/OS platform.

getIsMicrofiche

Method: getIsMicrofiche()

Returns: boolean. True if the document is in microfiche format. False otherwise. Applies only when connected to RSD Folders Server running on a z/OS platform.

getRawDocID

Method: getRawDocID()

Returns: byte[].Length, 64 bytes. The document id assigned to the document when it is created. This information is required when using the getFLDDocumentByID method. For more information, see "getFLDDocumentByID" on page 25.

getRunDate

Method: getRunDate()

Returns: java.util.Date. The run date and time.

getRunDateTime

Method: getRunDateTime()

Returns: String. The run date. Format: YYYYMMDDHHMM.

getRunNo

Method: getRunNo()

Returns: int. The run number.

getRunUserReference

Method: getRunUserReference()

Returns: String. The User Reference. Applies only when connected to RSD Folders Server running on a z/OS platform.

getStorageMedia

Method: getStorageMedia()

Returns: char. Indicates where the document is stored. Applies only when connected to RSD Folders Server running on a z/OS platform.

Possible values returned	Information
FLD_STORAGE_DIRECT	Direct access
FLD_STORAGE_NEAR	NearArchive
FLD_STORAGE_OAM	OAM
FLD_STORAGE_OPTICAL	Optical disk
FLD_STORAGE_PC	PC resident
FLD_STORAGE_TAPE	Tape or cartridge

isDocumentSigned

Method: isDocumentSigned()

Returns: boolean. True if the document has a valid digital signature. False otherwise.

Direct access to document information

An instance of the **FLDDocumentByID** interface is implemented when the method "getFLDDocumentByID", documented on page 25, has been executed.

For general information on using the document id, see "Unique document ID functionality" on page 20.

An instance of this interface:

- Can be referenced to perform actions to copy/move documents, transfer documents, delete documents.
- Can retrieve document information. It inherits the methods available for the FLDDocument interface, documented under the topic "General document information" on page 60.

Additional methods available for this interface are listed below.

Methods

createDocumentExtract

Implements an instance of the FLDExtract interface, which permits the extraction of the document. For more information, see "Extract a folder or document" on page 120. If the logged-on user is not authorized for this action, returns null. To execute this method, the isExtractAuthorized method must return True.

Method: createDocumentExtract()

isExtractAuthorized

Method: isExtractAuthorized()

Returns: boolean. True, the logged on user is authorized for this action. False, otherwise.

delete

Deletes the document.

Method: delete() **

getArchiveExpDate

Method: getArchiveExpDate()

Returns: java.util.Date. The archive expiration date.

getArchiveExpirationDate

Method: getArchiveExpirationDate()

Returns: String. The archive expiration date. Format: YYYYMMDD.

getDescriptorName

Method: getDescriptorName()

Returns: String. The descriptor name.

rename

The logged-on user must be authorized for this functionality. The isDocRenameAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Method: rename(String documentName) **

Parameter: "documentName" is the new name to assign to the document.

getRunUserReference

Method: getRunUserReference()

Returns: String. The User Reference. Applies only when connected to RSD Folders Server running on a z/OS platform.

getStorageMedia

Method: getStorageMedia()

Returns: char. Indicates where the document is stored. Applies only when connected to RSD Folders Server running on a z/OS platform.

Possible values returned	Information
FLD_STORAGE_DIRECT	Direct access
FLD_STORAGE_NEAR	NearArchive
FLD_STORAGE_OAM	OAM
FLD_STORAGE_OPTICAL	Optical disk
FLD_STORAGE_PC	PC resident
FLD_STORAGE_TAPE	Tape or cartridge

Copy or move a document

The FLDCopyMoveDocuments object facilitates the copying or moving of one or more documents from one folder to another. The logged-on use must be authorized for this functionality. The isDocMoveCopyAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Constructor: FLDCopyMoveDocuments(FLDDocumentList documentList) **

Parameter: "documentList" is a valid instance of the FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDCopyMoveDocuments(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Procedure

1. Execute the **setFromFolder** method to indicate the source folder that contains the document(s) to be copied or moved.
2. Execute the **setToFolder** method to indicate the target folder where the document(s) will be placed.
3. Create a **vector** of instances of the FLDDocumentListEntry interface that reference the documents to be copied or moved.
4. Execute the **copy** or **move** method to perform the operation.

Note

For the copy and move methods, instances of the FLDDocumentListEntry interface are required. For more information, see "FLDDocumentListEntry interface" on page 60.

For the setFromFolder and setToFolder methods, either an instance of the FLDFolderListEntry interface can be referenced or the folder name and folder type can be specified. For more information, see "Retrieve a list of filters" on page 37.

Methods

copy

Method: copy(java.util.Vector documentVector) **

Parameter: "documentVector" is a vector containing instances of the FLDDocumentListEntry interface that reference the documents to be copied.

getDocuments

Method: getDocuments()

Returns: java.util.Vector. Each element is an instance of the FLDDocumentListEntry interface.

getSourceFolderName

Method: getSourceFolderName()

Returns: String. The folder name as defined in the setFromFolder method.

getSourceFolderType

Method: getSourceFolderType()

Returns: int. The folder type as defined in the setFromFolder method.

getTargetFolderName

Method: getTargetFolderName()

Returns: String. The folder name as defined in the setToFolder method.

getTargetFolderType

Method: getTargetFolderType()

Returns: int. The folder type as defined in the setToFolder method.

move

Method: move(java.util.Vector documentVector) **

Parameter: "documentVector" is a vector containing instances of the FLDDocumentListEntry interface that reference the documents to be moved.

setFromFolder

There are two methods to set the source folder.

Method: setFromFolder(FLDFolderListEntry listEntry)

Method: setFromFolder(String folderName, int folderType)

Parameters:

- "listEntry" is an instance of the FLDFolderListEntry interface that references the source folder.
- "folderName" is a valid folder name within the folder type.
- "folderType" is the folder type.

setToFolder

There are two methods to set the target folder.

Method: setToFolder(FLDFolderListEntry listEntry)

Method: setToFolder(String folderName, int folderType)

Parameters:

- "listEntry" is an instance of the FLDFolderListEntry interface that references the target folder.
- "folderName" is a valid folder name within the folder type.
- "folderType" is the folder type.

Delete a document

The FLDDeleteDocument object facilitates the deletion of a document.

Constructor: FLDDeleteDocument(FLDDocumentList documentList) **

Parameter: "documentList" is a valid instance of FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDDeleteDocument(FLDQueryList queryList) **

Parameter: "queryList" is a valid instance of FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDDeleteDocument(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

deleteDocument

This method deletes a document. The logged-on user must have authorization to delete a document. The isDocDeleteAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.

Method: deleteDocument(FLDDocument document) **

Parameter: "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.

Manage notes

The **FLDNoteList** object facilitates retrieval, creation, modification and deletion of notes and note attachments. These objects are only valid for RSD Folders running on a Unix or Windows server platform for version 5.0 or higher.

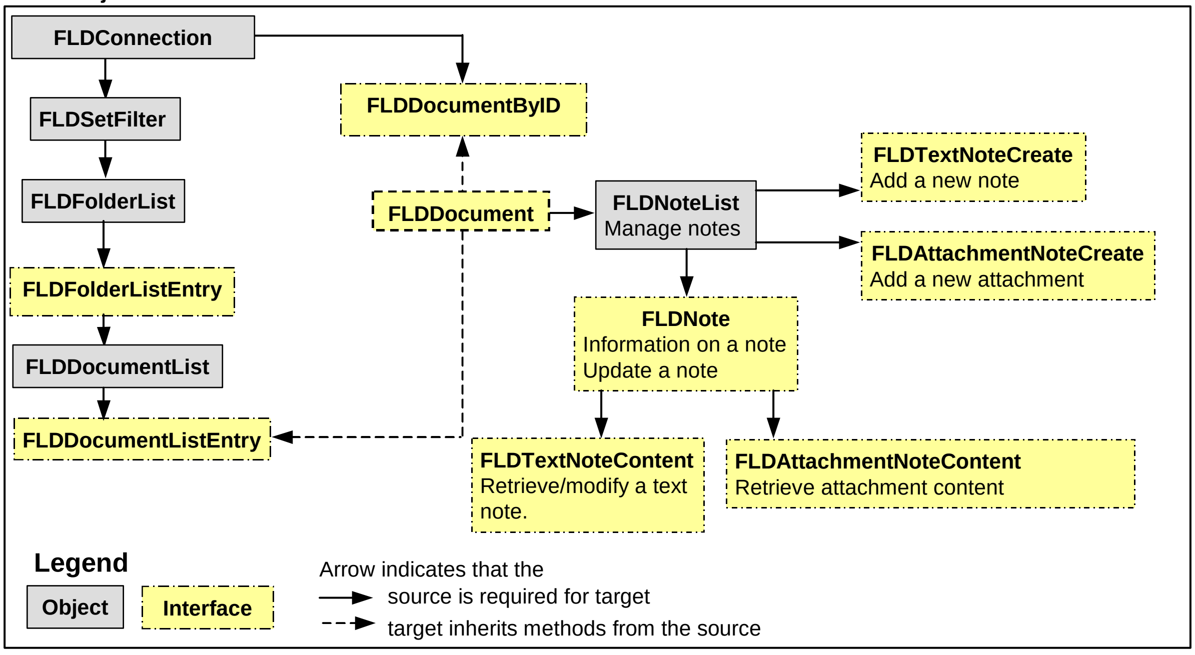
Types of notes

- Private - Can be retrieved, modified and deleted only by the creator of the note.
- Public - Can be created, retrieved, modified and deleted by anyone who can access the document.
- The end-user must be authorized for the above functionality. The necessary authorizations are noted in the documentation that follows.

Content of notes

- Text - Contains a description and text. Both can be modified or deleted.
- Attachment - Contains a description and a file attachment. The description can be modified. The file attachment cannot be modified. Instead, if the attachment is no longer applicable, delete it and, if necessary, add another attachment.

Note object and interfaces



Constructor

Constructor: (FLDDocument document) **

Parameter: "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.

Methods

createAttachmentNote

Implements an instance of FLDAttachmentNoteCreate. For more information, see "Create a note" on page 75.

Method: createAttachmentNote() **

createTextNote

Implements an instance of FLDTextNoteCreate. For more information, see "Create a note" on page 75.

Method: createAttachmentNote() **

Method: createTextNote() **

get and getList

To use these methods, the logged-on user must have authorization (isNoteViewAuthorized) to view a note. This information is returned in the FLDConnection object. For more information, see "Get information on user authorizations" on page 27.

Method: get(String noteID) **

Parameter: "noteID" is the note identifier, information that can be returned from the FLDNote interface.

Returns: An instance of FLDNote containing the information on the specified note.

Method: getList() **

Returns: A java.util.Iterator, which contains a list of notes associated with the document, private notes (those created by the logged-on user) and public notes. The next method returns an instance of FLDNote containing the information on the note in the list.

For more information, see "Retrieve information about a note" on page 72.

Retrieve information about a note

An instance of the FLDNote interface is implemented by executing commands get or getList in FLDNoteList.

An instance of the interface:

- Returns information on a note: author, modifier, whether the note is private or public, whether the note is text or attachment.
- Facilitates deletion of a note.
- Facilitates retrieval (and possible modification) or the contents of a note.
- Facilites modification of a note.

Method to delete a note

delete

Deletes the note. For a note that is Public, the logged-on user must have authorization (isNoteDeleteAuthorized). A private note can always be deleted. This information is returned in the FLDConnection object. See "Get information on user authorizations" on page 27.

To return information on whether the note is Private or Private, use the getContentAccess method documented in this topic.

Method: delete() **

Methods to retrieve information

getAuthor

Method: getAuthor()

Returns: String. The logon name of the user who created the note.

getContent

Returns an interface with the note content, which permits retrieval and modification of the content.

Method: getContent()

Returns: An instance of an interface with the note content. The interface used depends on the note content. To return information on whether the note is Text or Attachment, use the getType method documented in this topic.

- For Text, implements an instance of the FLDTextNoteContent interface. For more information, see "Modify a text note" on page 74.
- For Attachment, implements an instance of the FLDAttachmentNoteContent interface. For more information, see "Modify a text note" on page 74.

getContentAccess

Returns information on whether the note was created as private or public.

Method: getContentAccess()

Returns: String. ACCESS_CONTENT_PRIVATE, ACCESS_CONTENT_PUBLIC.

getCreationDate

Method: getCreationDate()

Returns: java.Util.Date. The note creation date.

getDescription

Method: getDescription()

Returns: String. The note description.

getID

Method: getID()

Returns: String. The note unique identifier that is assigned by the server.

getModificationAuthor

Method: getAuthor()

Returns: String. The logon name of the user who last modified the note.

getModificationDate

Method: getModificationDate()

Returns: java.Util.Date. The last note modification date.

getType

Returns whether the note content is Text or Attachment.

Method: getType()

Returns: int. NOTE_ATTACHMENT, NOTE_TEXT.

Methods to modify information

Use the set methods to modify the note description and/or access type. Use the update method to make the changes on the server.

setContentAccess

Facilitates making a private note --> public. To return information on whether the note is Private or Private, use the getContentAccess method documented in this topic.

Method: setContentAccess(String contentAccessType) **

Parameter: "contentAccessType" is the string ACCESS_CONTENT_PUBLIC.

setDescription

Modifies the note description.

Method: setDescription(String noteDescription)

Parameter: "noteDescription" is the description of the note.

update

For a note that is Public, the logged-on user must have authorization (isNoteModifyAuthorized). A private note can always be modified. This information is returned in the FLDConnection object. See "Get information on user authorizations" on page 27.

Method: update() **

Modify a text note

The getContent method (FLDNote) implements an instance of the **FLDTextNoteContent** interface, which facilitates retrieval and modification of the note content.

For a text note that is Public, the logged-on user must have authorization (isNoteModifyAuthorized). A private note can always be modified. This information is returned in the FLDConnection object. See "Get information on user authorizations" on page 27.

Methods

getMimeContentType

Method: getMimeContentType()

Returns: String. The mime content type of the text note.

getText

Method: getText() **

Returns: String. The content of the text note.

setText

Method: getText(String text)

Parameter: "text" is the replacement text for the note.

Retrieve an attachment

The getContent method (FLDNote) implements an instance of the **FLDTextNoteContent** interface, which facilitates retrieval of the attachment content.

Methods

getContentLength

Method: getContentLength()

Returns: long. The length (in bytes) of the attachment data.

getFilePath

Method: getFilePath()

Returns: String. The original path of the attachment file.

getMimeContentType

Method: getMimeContentType()

Returns: String. The mime type of the attachment content.

transfer

Transfers the attachment content to the specified output stream.

Method: transfer(java.io.OutputStream outputStream) **

Parameter: "outputStream" is the stream used to receive the data.

Create a note

The createTextNote method (FLDNoteList) implements an instance of the **FLDTextNoteCreate** interface, which facilitates the creation of a private or public text note.

The createAttachmentNote method (FLDNoteList) implements an instance of the **FLDAttchmentNoteCreate** interface, which facilitates the creation of a private or public attachment.

The logged-on user must have authorization (isNoteAddAuthorized). This information is returned in the FLDConnection object. See "Get information on user authorizations" on page 27.

Methods

Unless specified, the methods listed below are available in both interfaces.

add

Creates a new note on the server using the information defined in the following set methods.

Method: add() **

setContentAccess

Method: setContentAccess(String contentAccessType) **

Parameter: "contentAccessType" is the access type: private or public.

Permissible values: ACCESS_CONTENT_PRIVATE or ACCESS_CONTENT_PUBLIC.

setContentData

These methods are only available in the FLDAttachmentNoteCreate interface.

Method: setContentData(java.io.File attachmentFile, String mimeType) **

Method: setContentData(java.io.InputStream inputStream, String filePath, long contentLength, String mimeType) **

Parameters: "contentAccessType" is the access type: private or public.

- "attachmentFile" is full path and file name of the attachment.

  If the attachment is in PDF format, it must not be secured using Adobe or compatible security.

  Restriction for the file extension:

  - Maximum length is 3 characters.
  - If the extension is exactly 3 characters, it cannot contain any numbers.
- "mimeType" is the mime type of the attachment file.
- "inputStream" contains the content.
- "filePath" is the information on the data source file.

  "contentLength" is the length of the content in bytes.

setDescription

Method: setDescription(String noteDescription)

Parameter: "noteDescription" is the description for the note.

setText

This method is only available in the FLDTextNoteCreate interface.

Method: setText(String text)

Parameter: "text" is the content of the text note.

Manage document versions

The **FLDVersionList** object facilitates the viewing of versions, the addition of new versions and deletion of existing versions. This object and related interfaces for Version Management are only valid for RSD Folders running on a Unix or Windows server platform for version 5.0 or higher.

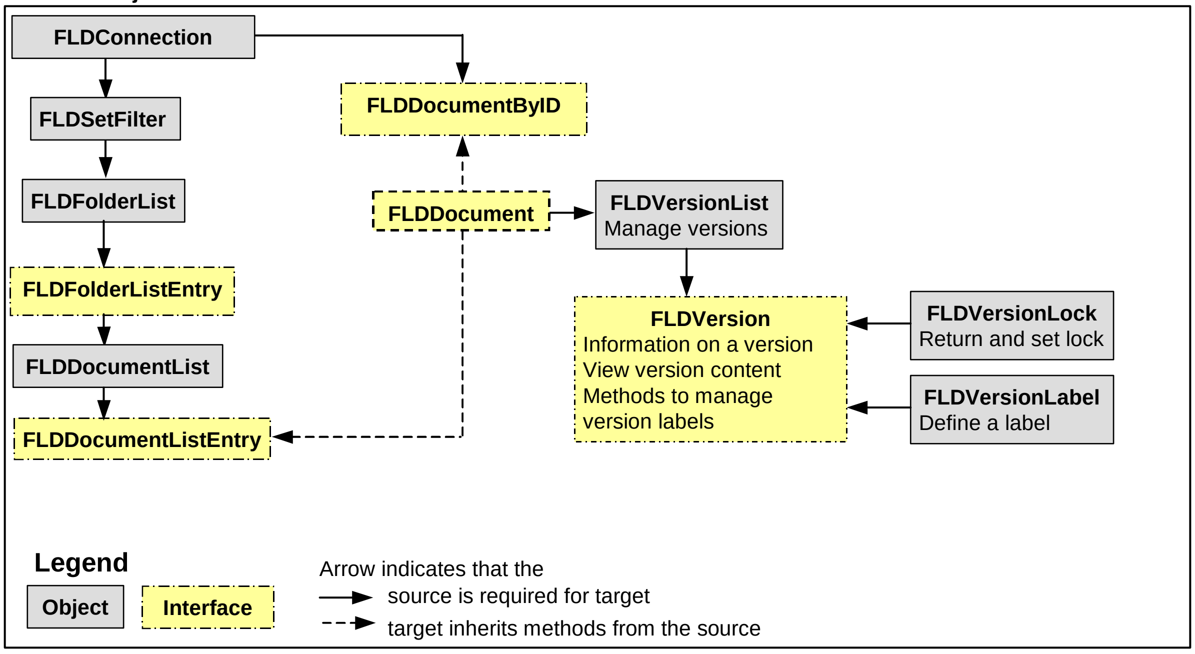
The logged-in user must have specific authorizations to use this functionality. The necessary authorizations are noted in the documentation that follows.

The active (latest) version can be:

- Locked and unlocked.
- Can be checked out. This permits modification of the version and automatically locks the checked out version.
- Can be checked in. This permits the inclusion of a new version, which will then become the latest version. It can also automatically unlock the version that was previously checked-out and locked.

In addition, labels can be attached to a version to help identify it.

Version object and interfaces



Constructor

Constructor: (FLDDocument document) **

Parameter: "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.

Methods

To use the checkIn, checkOut, lock, and unLock methods, the logged-on user must have authorization (isVersionAddAuthorized) to view a version. This information is returned in the FLDConnection object. For more information, see "Get information on user authorizations" on page 27.

checkIn

These methods permit the insertion, in the RSD Folders database of a new version of the document referenced by the constructor.

Method: checkIn(java.io.File fileToUpload, String description, boolean keepVersionLocked) **

Method: checkIn(java.io.InputStream inputStream, String filePath, String description, boolean keepVersionLocked) **

Parameters:

- "fileToUpload" is the full path and filename of the file to upload.
- "inputStream" contains the content.
- "filePath" is the full path and filename of the file to upload.
- "description" is a description for the new version. Optional.
- "keepVersionLocked" - If set to true, the document remains locked. If set to false, the document is unlocked.

checkOut

Locks the document and copies the content of the active (latest) document version to the outputStream parameter. If the first method is used, the RSD Folders server will provide the lock ID. The second method is available if using a protocol that furnishes a unique lock ID.

Method: checkOut(java.io.OutputStream output) **

Method: checkOut(java.io.OutputStream output, String lockID, String lockSubject, String path, String encoding) **

Parameters:

- "output" is the output stream where the content is copied.
- "lockID" is the unique lock ID. If Null, the RSD Folders server generates the lock ID.
- "lockSubject" is the lock subject. Optional.
- "path" is the path where the content is copied. Optional.
- "encoding" is the encoding to be applied to the content. Optional. Default: encoding used for the Java Virtual machine.

deleteVersion

Deletes the specified version. To use this method, the logged-on user must have authorization (isVersionDeleteAuthorized). This information is returned in the FLDConnection object. For more information, see "Get information on user authorizations" on page 27.

Method: deleteVersion(FLDVersion version) **

Parameter: "version" is a valid instance of the FLDVersion interface. For more information, see "Manage a specific version of a document" on page 78.

getActiveVersion, getList and getVersion

To use these methods, the logged-on user must have authorization (isVersionViewAuthorized) to view a version. This information is returned in the FLDConnection object. For more information, see "Get information on user authorizations" on page 27.

Method: getActiveVersion() **

Returns: A valid instance of the FLDVersion interface containing methods to work with the latest version of the document.

Method: getList() **

Returns: A java.util.Iterator, which contains a list of versions for the document with the latest version listed first. The next method returns an instance of FLDVersion containing methods to work with a version.

Method: getVersion(String versionID) **

Parameter: "versionID" is the version number.

Returns: A valid instance of the FLDVersion interface containing methods to work with the specified version of the document.

For more information, see "Manage a specific version of a document" on page 78.

isLocked

Method: isLocked() **

Returns: boolean. True if the document is locked. False otherwise.

lock

Locks the document. Note that if the method checkOut is used, the document is automatically locked and this method need not be used. The logged-on user must have authorization (isVersionAddAuthorized).

Method: lock() **

Method: lock(String lockSubject, String lockID, String path) **

Parameters:

- "lockID" is the unique lock ID. If Null, the RSD Folders server generates the lock ID.
- "lockSubject" is the lock subject. Optional.
- "path" is the path where the downloaded document is located when calling lock method from checkOut action. Optional.

unlock

Unlocks a document. Note that the document may be unlocked automatically when a version is checked in, depending on the values used for the checkIn method parameters. The logged-on user must have authorization (isVersionAddAuthorized).

Method: unlock() **

Manage a specific version of a document

An instance of the **FLDVersion** interface is implemented by executing commands getActiveVersion, getVersion or getList (next method) in FLDVersionList.

An instance of the interface:

- Returns information on a version.
- Facilitates management of labels for the version. See 'Label methods'
- Facilities modification of a description. See 'Modify description method'

Note that the getInternalDocument method is for internal use and not documented.

Retrieve information methods

The logged-on user must have authorization (isVersionViewAuthorized).

getAuthor

Method: getAuthor()

Returns: String. The name of the user that inserted the version.

getContent

Method: getContent(java.io.OutputStream output, String encoding) **

Parameters:

- "output" is the output stream where the content is copied.
- "encoding" is the encoding to be applied to the content. Optional. Default: encoding used for the Java Virtual machine.

Returns: The document version content.

getDate

Method: getDate()

Returns: java.util.Date. The date that the document version was created.

getDescription

Method: getDescription()

Returns: String. The description for the document version.

getLock

Method: getLock()

Returns: A valid instance of FLDVersionLock that contains methods to retrieve information on the lock. Null if there is no lock. For more information, see "Manage document locks" on page 81.

getRevision

Method: getRevision()

Returns: String. The revision number for the document version.

isActiveVersion

The active version is the latest version of the document.

Method: isActiveVersion()

Returns: boolean. True if this version is the active one, false otherwise.

Modify description methods

To modify a document version description, both methods must be executed. The logged-on user must have authorization (isVersionAddAuthorized).

setDescription

Method: setDescription(String description)

Parameter: "description" is the description for the document version.

update

Updates the document version with the description specified by the setDescription method.

Method: update() **

Label methods

The logged-on user must have authorization (isVersionAddAuthorized). For more information, see "Manage version labels" on page 80.

addLabel

Creates a label for the document version.

Method: addLabel(FLDVersionLabel label) **

Parameter: "label" is a valid instance of the FLDVersionLabel interface containing the label to add to the list of labels associated with the document version.

deleteLabel

Removes a label associated with a version of a document.

Method: deleteLabel(FLDVersionLabel label) **

Method: deleteLabel(String labelName) **

Parameters:

- "label" is a valid instance of the FLDVersionLabel object containing the name of the label to remove from the list of labels associated with the document version.
- "labelName" is the name of the label that should be removed.

getLabelList

Method: getLabelList() **

Returns: A java.util.Iterator, which contains a list of labels associated with the document version or null if there are no labels. The next method returns an instance of FLDVersionLabel containing methods to work with a label.

Manage version labels

An instance of the **FLDVersionLabel** object facilitates the management of labels associated with a version of a document.

Notes:

- The logged-on user must have authorization (isVersionAddAuthorized).
- The method toString is for internal use only and not documented.

Constructor: FLDVersionLabel()

Constructor: FLDVersionLabel(String labelValue, boolean floatingLabel)

Parameters:

- "labelValue" is the name of the label.
- "floatingLabel" If true, then the label is automatically always associated with the active (latest) version of the document. If false, then the label is always associated with a specific version of the document.

Methods

getLabelValue

Method: getLabelValue()

Returns: String. The name of the label.

isFloatingLabel

Method: isFloatingLabel()

Returns: boolean. True if the label is always associated with the active (latest) version of the document. False otherwise.

setFloatingLabel

Method: setFloatingLabel(boolean floatingLabel)

Parameter: See "floatingLabel" documented above.

setLabelValue

Method: setLabelValue(String labelValue)

Parameter: See "labelValue" documented above.

Manage document locks

An instance of the **FLDVersionLock** object facilitates the management of document locks.

Notes:

- The logged-on user must have authorization (isVersionAddAuthorized).
- The method toString is for internal use only and not documented.

Constructor: FLDVersionLock()

Set methods

setDate

Method: setDate(java.util.Date date)

Parameter: "date" is the date that the document is locked.

setLockedBy

Method: setLockedBy(String lockedBy)

Parameter: "lockedBy" is the name of user who is locking the document.

setPath

Method: setPath(String path)

Parameter: "path" is the path where the content is copied. Optional.

setSubject

Method: setSubject(String subject)

Parameter: "subject" is the lock subject. Optional.

setUUID

Method: setUUID(String UUID)

Parameter: "UUID" is the unique lock ID. If Null, the RSD Folders server generated the lock ID.

Get methods

getDate

Method: getDate()

Returns: java.util.Date. The lock date.

getLockedBy

Method: getLockedBy(lockedBy)

Returns: String. The name of user who locked the document.

getPath

Method: getPath(String path)

Returns: String. The path where the content is copied.

getSubject

Method: getSubject(String subject)

Returns: String. The lock subject.

getUUID

Method: getUUID(String UUID)

Returns: String. The unique lock ID.

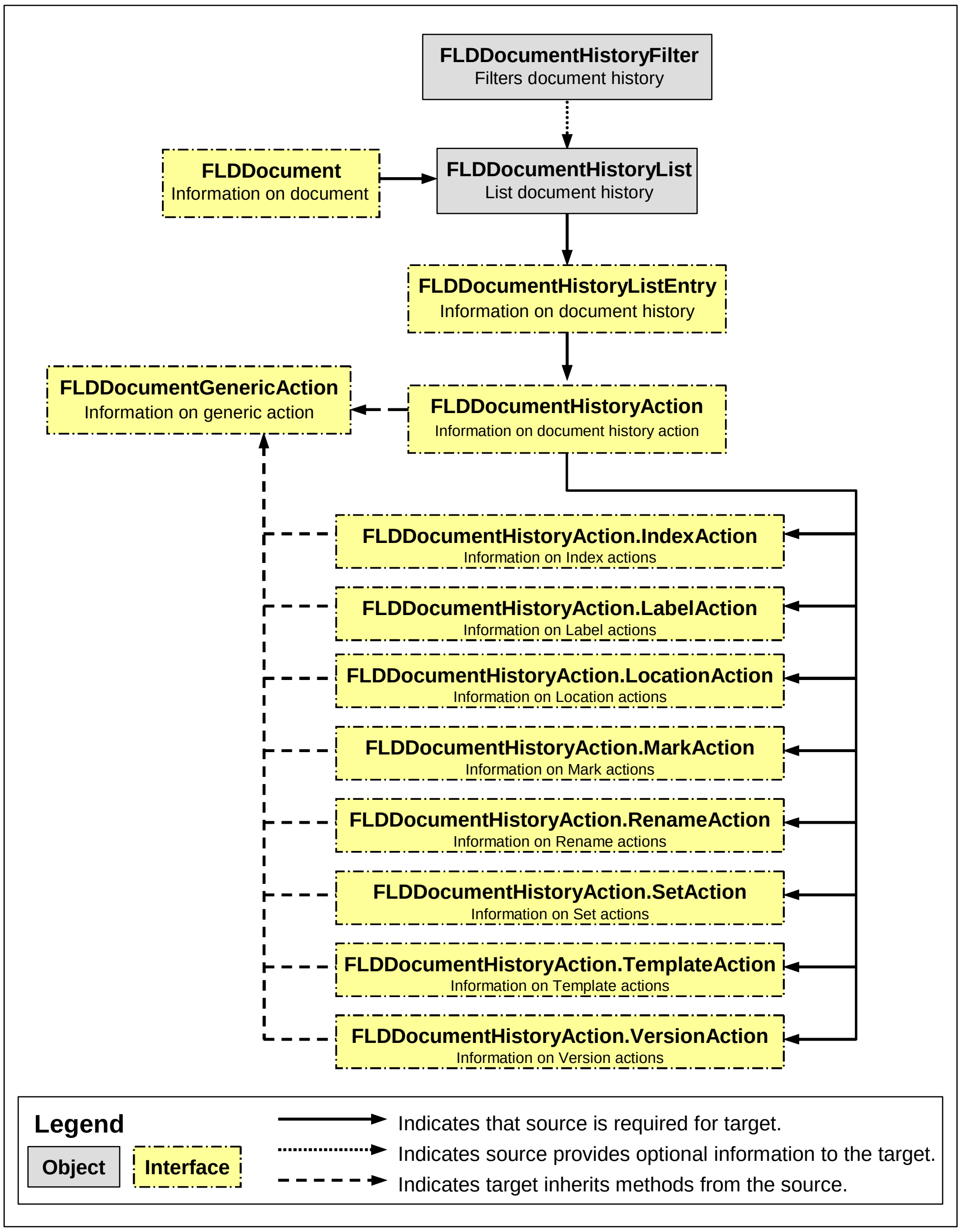
Manage document history

The **FLDDocumentHistoryList** object facilitates the viewing of a list of actions performed on a document. This object and related interfaces for Document History are only valid for RSD Folders version 5.1 or higher running on a Unix or Windows server platform.

The logged-in user must have specific authorizations to use this functionality.

The necessary authorizations are noted in the documentation that follows.

Document History object and interfaces



FLDDocumentHistoryFilter
Filters document history
FLDDocument
Information on document
FLDDocumentHistoryList
List document history
FLDDocumentHistoryListEntry
Information on document history
FLDDocumentGenericAction
Information on generic action
FLDDocumentHistoryAction
Information on document history action
FLDDocumentHistoryAction.IndexAction
Information on Index actions
FLDDocumentHistoryAction.LabelAction
Information on Label actions
FLDDocumentHistoryAction.LocationAction
Information on Location actions
FLDDocumentHistoryAction.MarkAction
Information on Mark actions
FLDDocumentHistoryAction.RenameAction
Information on Rename actions
FLDDocumentHistoryAction.SetAction
Information on Set actions
FLDDocumentHistoryAction.TemplateAction
Information on Template actions
FLDDocumentHistoryAction.VersionAction
Information on Version actions
Legend
Object
Interface
Indicates that source is required for target.
Indicates source provides optional information to the target.
Indicates target inherits methods from the source.

Retrieve information methods

The logged-on user must have authorization (isPrivateDocumentHistoryAuthorized) or (isPublicDocumentHistoryAuthorized).

isPrivateDocumentHistoryAuthorized

Method: isPrivateDocumentHistoryAuthorized()

Returns: boolean. True, if the user is authorized to see private document history. False, otherwise.

isPublicDocumentHistoryAuthorized

Method: isPublicDocumentHistoryAuthorized()

Returns: boolean. True, if the user is authorized to see public document history. False, otherwise.

Retrieve document history list

The **FLDDocumentHistoryList** object returns the document history as a list. The document history list is either private or public. It lists the actions made on the document.

The list returned also depends on a FLDDocumentHistoryFilter object to filter the entries of the list. See "Manage document history filter" on page 84.

Document history should be supported by the server. The ability to list the private history and the public history of the document depends on the user authorizations.

Usage: After the creation of the FLDDocumentHistoryList object, the readFirst or the readLast method should be called first. Then the readPrevious and the readNext enable you to navigate in the list.

Constructor: FLDDocumentHistoryList(FLDDocument document)

Constructor: FLDDocumentHistoryList(FLDDocument document, FLDDocumentHistoryFilter filter)

Parameters:

- "document" is a valid instance of the FLDDocument interface and the document to retrieve the history from.
- "filter" is a valid instance of the FLDDocumentHistoryFilter object to filter the list by userName. If null, the document history list is not filtered. The filter is used only if the user is authorised to list public history.

Notes:

- The method toString is for internal use only and not documented.

Methods

readFirst

Reads matching entries from the server starting with the first one. The first entry is the first event on the document (the oldest one).

Method: readFirst()**

Returns: boolean. True if there are occurences available.

readPrevious

Reads matching entries from the server before the current one.

Method: readPrevious()**

Returns: boolean. True if there are occurences available.

readNext

Reads matching entries from the server after the current one.

Method: readNext()**

Returns: boolean. True if there are occurences available.

readLast

Reads matching entries from the server starting with the last one. The last entry is the last event on the document (the most recent one).

Method: readLast()**

Returns: boolean. True if there are some occurences available.

close

Notifies the server to stop processing for this object. This method should always be called after the document history list has been retrieved.

Method: close()**

getEntry

Usage: a read method must be executed before. The readFirst, readPrevious, readNext, and readLast methods return a boolean indicating if an entry is available. The returned entry becomes the current entry of the list.

Method: FLDDocumentHistoryListEntry getEntry()**

Returns: An instance of the FLDDocumentHistoryListEntry interface with information contained in the record.

getTotalOccurences

Returns the number of occurences matching the filter on the server.

Method: getTotalOccurences()

Returns: int.

Manage document history filter

The **FLDDocumentHistoryFilter** object filters the history list of a document. This filter is applied to the user name.

When the user is authorized to see the public document history, he can filter the list with a user name. The connected user must be authorized to see public document history to filter on the user name.

When the user is authorized to see only his private history, he can see only his own actions.

Constructor: FLDDocumentHistoryFilter()

Constructor: FLDDocumentHistoryFilter(String userName)**

Parameters:

- "userName" is the user name to be used by the filter.

Methods

getUserName

Returns the userName used by the filter.

Method: getUserName()

Returns: String. Returns null if no username is used.

setUserName

The userName to be used by the filter.

Method: setUserName(String userName)**

Parameters:

- "userName" is the user name to be used by the filter.

Retrieve document history list entry

The **FLDDocumentHistoryListEntry** returns an entry of the FLDDocumentHistoryList list. An entry of the list has: a unique Id, a date, a userName, and an action. The Id is used to navigate through the list.

Methods

getId

The Id of the document history entry. The Id is used to navigate through the list.

Method: getId

Returns: int. The Id is between 1 and the number of occurences in the list.

getUserName

The userName associated to the document history entry. This is the userName of the user who performed the action.

Method: getUserName()

Returns: String.

getDate

The date of the action.

Method: getDate()

Returns: Date.

getAction

The action. For more information on actions see following item.

Method: getAction()

Returns: FLDDocumentHistoryAction

getActionLabel

The action label. A translated sentence which summarizes the action. The action label is translated into the language specified for the connected user.

Method: getActionLabel()

Returns: String

Retrieve generic information on document history actions

The **FLDDocumentHistoryGenericAction** interface gives general information on actions made on documents. Every action has an actionType and an actionLabel.

Some actions also have some specific data. This is the case for the actions related to: Marks, ABI Indexes, Locations of documents, Templates, Labels, Versions, Rename of documents, Extractions by set.

The methods contained in the FLDDocumentHistoryGenericAction interface are inherited by instances of the following interfaces: FLDDocumentHistoryAction, FLDDocumentHistoryAction.IndexAction, FLDDocumentHistoryAction.LabelAction, FLDDocumentHistoryAction.VersionAction, FLDDocumentHistoryAction.LocationAction, FLDDocumentHistoryAction.MarkAction, FLDDocumentHistoryAction.RenameAction, FLDDocumentHistoryAction.SetAction and FLDDocumentHistoryAction.TemplateAction

Methods

getActionType

The action type

Method: getActionType()

Returns: String.

getActionClass

The java action class representing the action with its specific data.

Method: getActionClass()

Returns: String.

Possible values are: FLDDocumentHistoryAction, FLDDocumentHistoryAction.IndexAction, FLDDocumentHistoryAction.LabelAction, FLDDocumentHistoryAction.VersionAction, FLDDocumentHistoryAction.LocationAction, FLDDocumentHistoryAction.MarkAction, FLDDocumentHistoryAction.RenameAction, FLDDocumentHistoryAction.SetAction and FLDDocumentHistoryAction.TemplateAction

getActionLabel

The action label. A translated sentence which summarizes the action. e.g. document deletion.

Method: getActionLabel()

Returns: String.

Retrieve information on Index actions

The **FLDDocumentHistoryAction.IndexAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The IndexAction interface gives information on an action made on a ABI index. This is used when an ABI index is added, updated or deleted.

Methods

getFieldName

The name of the index field descriptor.

Method: getFieldName()

Returns: String.

getFieldValue

The index field value.

Method: getFieldValue()

Returns: String.

Retrieve information on label actions

The **FLDDocumentHistoryAction.LabelAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The LabelAction interface gives you information on an action where a label is involved. This is used when a document label is added, or deleted.

Methods

getLabel

The label of the document.

Method: getLabel()

Returns: String.

Retrieve information on document location actions

The **FLDDocumentHistoryAction.LocationAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The LocationAction interface gives information on an action where a document is moved or copied. The location is defined by a folder and a document.

Methods

getFolderName

The name of the folder where the document is moved or copied to.

Method: getFolderName()

Returns: String. The folder name.

getFolderType

The type of the folder where the document is moved or copied to. This folder type is an integer between 1 and 998.

Method: getFolderType()

Returns: String. The folder type

getFolderTypeDescription

The folder type description of the folder where the document is moved or copied to.

Method: getFolderTypeDescription()

Returns: String. The folder type description

getDocumentName

The name of the document which is copied or moved.

Method: getDocumentName()

Returns: String. The document name.

getDocumentType

The document type of the document which is copied or moved. This document type is an integer between 1 and 998.

Method: getDocumentType()

Returns: String. The document type.

getDocumentTypeDescription

The type description of the document which is moved or copied.

Method: getDocumentTypeDescription()

Returns: String. The document type description.

Retrieve information on document mark actions

The **FLDDocumentHistoryAction.MarkAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The MarkAction interface gives you information on an action made on a mark. This is used when a mark is modified.

Methods

getMarkId

The mark Id of the action.

Method: getMarkId()

Returns: String.

getMarkStatus

The mark status of the action.

Method: getMarkStatus()

Returns: boolean.

getMarkDesc

The mark description of the action. The mark description is associated to the mark Id.

Method: getMarkDesc()

Returns: String.

getMarkLabel

The mark label of the action. The mark label is associated to the mark status.

Method: getMarkLabel()

Returns: String.

Retrieve information on document rename actions

The **FLDDocumentHistoryAction.RenameAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The RenameAction interface gives information on a document rename action.

Methods

getOldName

The old document name.

Method: getOldName()

Returns: String.

getNewName

The new document name.

Method: getNewName()

Returns: String.

Retrieve information on document extraction actions

The **FLDDocumentHistoryAction.SetAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The SetAction interface gives information on a document extraction action. For example, when a document is extracted in a set of documents.

Methods

getName

The name of the set.

Method: getName()

Returns: String.

Retrieve information on document template actions

The **FLDDocumentHistoryAction.TemplateAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The TemplateAction interface gives information on an action where a template is involved. This is used when a document is transformed with a template defined in the RSD Folders Thin Client.

Methods

getTemplate

The name of the template used for the document transformation.

Method: getTemplate()

Returns: String.

Retrieve information on document version actions

The **FLDDocumentHistoryAction.VersionAction** interface inherits methods from the FLDDocumentHistoryGenericAction interface.

The VersionAction interface gives information on an action where a version is involved. This is used when a document is checked in or checked out. This is also used when a documnent version is added, viewed, or deleted.

Methods

getVersion

The version of the document.

Method: getVersion()

Returns: String.

Retrieve information on document history actions

The **FLDDocumentHistoryAction** interface implements FLDDocumentHistoryGenericAction.

Methods

getIndexAction

An object that implements FLDDocumentHistoryAction.IndexAction.

Method: getIndexAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.IndexAction.

getLabelAction

An object that implements FLDDocumentHistoryAction.LabelAction

Method: getLabelAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.LabelAction.

getLocationAction

An object that implements FLDDocumentHistoryAction.LocationAction.

Method: getLocationAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.LocationAction.

getMarkAction

An object that implements FLDDocumentHistoryAction.MarkAction.

Method: getMarkAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.MarkAction.

getRenameAction

An object that implements FLDDocumentHistoryAction.RenameAction

Method: getRenameAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.RenameAction.

getSetAction

An object that implements FLDDocumentHistoryAction.SetAction

Method: getSetAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.SetAction.

getTemplateAction

An object that implements FLDDocumentHistoryAction.TemplateAction

Method: getTemplateAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.TemplateAction.

getVersionAction

An object that implements FLDDocumentHistoryAction.VersionAction

Method: getVersionAction()

Returns: Null is returned if is not an object implementing FLDDocumentHistoryAction.VersionAction.

Query objects

Section overview

Query objects are used to access Advanced Business Index (ABI) functionality.

These objects are only valid for RSD Folders running on a Unix or Windows server platform where ABI functionality has been implemented.

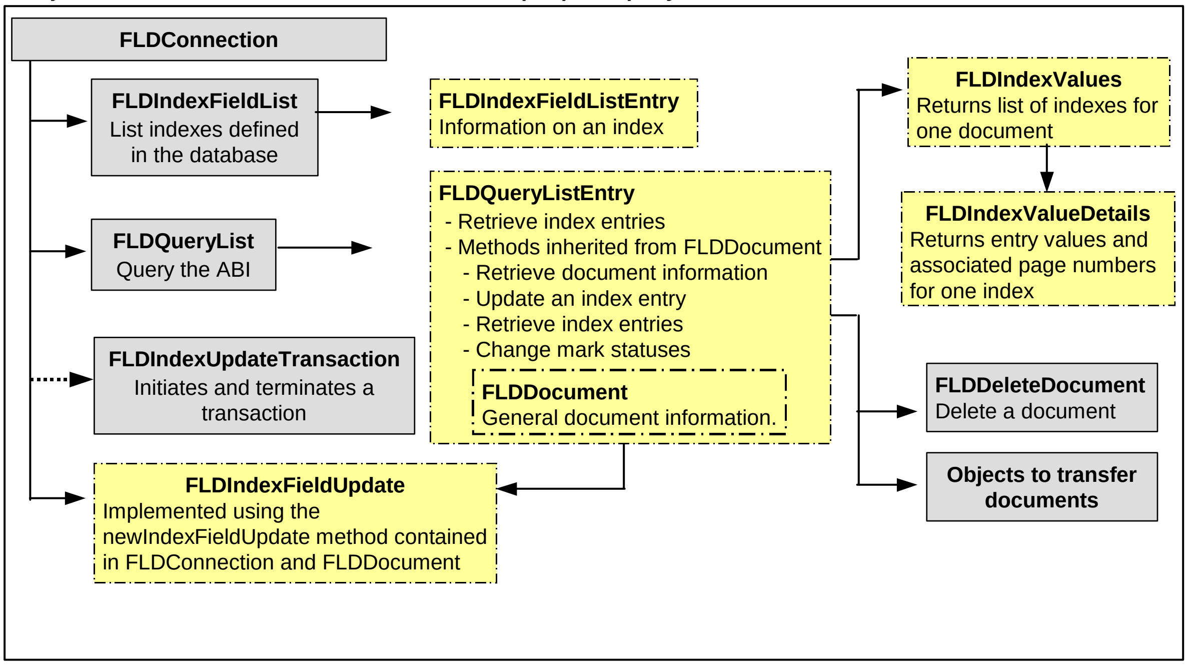
Warning: The DB Server must be active to use these objects.

Below is a list of the objects used to retrieve a list of index field descriptors, to retrieve a list of documents that match an index query and to retrieve information on a specific document.

Objects covered in this section:

- "Retrieve a list of index field descriptors" on page 92 (FLDIndexFieldList)
- "Retrieve a list of documents based on an index query" on page 95 (FLDQueryList)
- "Get information on the indexes" on page 102 (FLDIndexValues and FLDIndexValueDetails))
- "Add, delete or change index entries" on page 103 (FLDIndexFieldUpdate)
- "Index Update transaction" on page 106 (FLDIndexUpdateTransaction)

Objects related to the Advanced Business Index (ABI) and query list


FLDConnection
FLDIndexFieldList
List indexes defined in the database
FLDQueryList
Query the ABI
FLDIndexUpdateTransaction
Initiates and terminates a transaction
FLDIndexFieldUpdate
Implemented using the newIndexFieldUpdate method contained in FLDConnection and FLDDocument
FLDIndexFieldListEntry
Information on an index
FLDQueryListEntry
- Retrieve index entries
- Methods inherited from FLDDocument
- Retrieve document information
- Update an index entry
- Retrieve index entries
- Change mark statuses
FLDDocument
General document information.
FLDIndexValues
Returns list of indexes for one document
FLDIndexValueDetails
Returns entry values and associated page numbers for one index
FLDDeleteDocument
Delete a document
Objects to transfer documents

Retrieve a list of index field descriptors

The **FLDIndexFieldList** object provides a list of index field descriptors in ascending alphabetic order.

Constructor: FLDIndexFieldList(FLDConnection connectObject) **

Parameter: "connectObject" is a valid instance of the FLDConnection object.

Access a record in the list

The following methods navigate forward in a list of index field descriptors.

Once a 'read' method has been executed, the next method retrieves the information for that entry and moves to the next record. To verify the existence of more records, use the 'hasNext' method.

hasNext

Method: hasNext()

Returns: boolean. True, if the next method can be used to retrieve information on an entry. False, otherwise.

next

Retrieves information on a record and then moves to the next record. The information is returned to an instance that implements the FLDIndexField interface.

Method: next() **

readFirst

Positions at the first record in the list.

Method: readFirst() **

readNext

Positions at the next record in the list. Assumes a readFirst has been executed.

Method: readNext() **

readPosition

Positions at the record that contains the index field descriptor specified by the fieldName parameter, or if the field name is not found, positions to the next field name in the list.

Method: readPosition(String fieldName) **

Parameter: "fieldName" is the name of the index field to be accessed, maximum 10 characters.

Get the information in the record

When the next method is executed, the following methods retrieve information stored in an instance that implements the **FLDIndexFieldListEntry** interface.

Note on returning information

The getFieldInformation returns all information on the field, concatenated. Therefore, it is suggested that you use the other methods to return specific information.

For example:

- Use the isDate method to return whether the field is a date. If so, use the getFormat method to return the format for the date.
- Use the isList method to return whether the field is constrained to permit only a specific list of values. If so, use the getListKey and getListDescription methods to return the list values and the corresponding list descriptions.

getFormat

Method: getFormat()

Returns: String. If the index field is constrained to a specific format (as is normally the case for the date field type), returns the format for the index field descriptor. Otherwise, returns null.

getFieldDescription

Method: getFieldDescription()

Returns: String. The index field description, maximum 40 characters.

getFieldInformation

Method: getFieldInformation()

Returns: String. Information on the definition including: format of values, type of constraint, minimum and maximum range, value list, folder and document type restrictions.

getFieldName

Method: getFieldName()

Returns: String. The name of the index field descriptor.

getFieldSize

Method: getFieldSize()

Returns: int. Only for String formatted data, returns the maximum number of characters in the string. Otherwise, returns null.

getFieldType

Method: getFieldType()

Returns: String. The format of the data indexed. For more information, see "Index field types" on page 94.

getListDescription

Method: getListDescription ()

Returns: java.util.ArrayList. If the index field is constrained to a list of values, returns the descriptions for each list item. Otherwise, returns null.

getListKey

Method: getKeyList()

Returns: java.util.ArrayList. If the field is constrained to a list of values, returns the list of values. Otherwise, returns null.

isDate

Method: isDate()

Returns: boolean. True, the index field has type date. False, otherwise.

isList

Method: isList()

Returns: boolean. True, the index field has a defined list of permissible values. False, otherwise.

Index field types

Index entries must always conform to the format defined for the specific index field descriptor on the RSD Folders server. However, the actual format for storing the data on the server is different than the format used to return index entries to the API or to send updated index entries to the server. Where the format is different, the Key serves to manage the format.

Explanation of the chart following:

- The **Field Type** column indicates the string used to specify the field type.
- The **RSD Folders server** column indicates the format defined for the index field descriptor on the server.
- The **Java API** column indicates the API format for retrieving or sending entries

Following is a list of index field types.

Field type	RSD Folders server index field format	RSD Folders Java API format for retrieving or sending entries
B	Boolean	Boolean
D	Date	Date
S	String	String
I	Integer	Integer
l	Long	Integer
u	Unsigned long	Long
s	Short	Short
v	Unsigned short	Integer
R	Real	Float
f	Float	Float
d	Double	Double

Retrieve a list of documents based on an index query

The **FLDQueryList** object queries index entries associated with one or multiple indexes and retrieves a list of documents that match the search criteria. The list of documents is sorted by Folder Type and then by document creation (capture) date/time.

Constructor: FLDQueryList(FLDConnection connectObject) **

Constructor: FLDQueryLIst(FLDSetFilter setFilter) **

Parameters:

- "connectObject" is a valid instance of the FLDConnection object.
- "setFilter" is a valid FLDSetFilter object. For more information, see "Set a filter" on page 38.

Notes:

- The logged user name must authorized to perform an index search. The isIndexSearchAuthorized method must return true. For more information on authorizations, see "Get information on user authorizations" on page 27.
- An instance of FLDQueryList can be used as the referenced object to get information on a document, transfer documents and access a list of document resources.

Query results returned by the server

When the server executes a query, it initially builds a list of all documents (as explained following). It then returns a subset of this initial list, which depends on the filtering applied.

The default is to return only documents that meet the criteria in the active F/D filter.

However, it is possible to change the filtering criteria without changing the active F/D filter.

To change the filtering criteria, the method "enableFilterList" documented on page 99 must be executed. When executed (set to True) the following occurs:

- The server immediately sets the filtering to use all F/D filters defined for the logged-on user. At that point, the query results will include all documents that meet the criteria in all F/D filters authorized for that user.
- The method "addToFilterList" documented on page 98 becomes available to specify which of those filters should be used to filter the query results. The first time that the method is executed, the list of filters is changed to only contain the filter name specified by this method. Additional filters can be added to the list be executing the method multiple times.
- The method "getFilterName" documented on page 101 becomes available to retrieve information on the filter name used to include the document in the query results.

Query format

The query itself must be entered in post-fixed notation. Rules for using "post-fixed" notation:

- The criteria (consisting of the index field to search, the comparison operator and the value to compare) are pushed first, followed by the operator (the Boolean connector).
- When a Boolean connector is found, it looks for the two previous sets of criteria (not yet used) or if only one set is found, the result of another operation previously performed.
- The order, in which sets of criteria and Boolean connectors are sent, determines how the query is executed.

"Post-fixed processing" on page 96 shows different configurations for executing the same query.

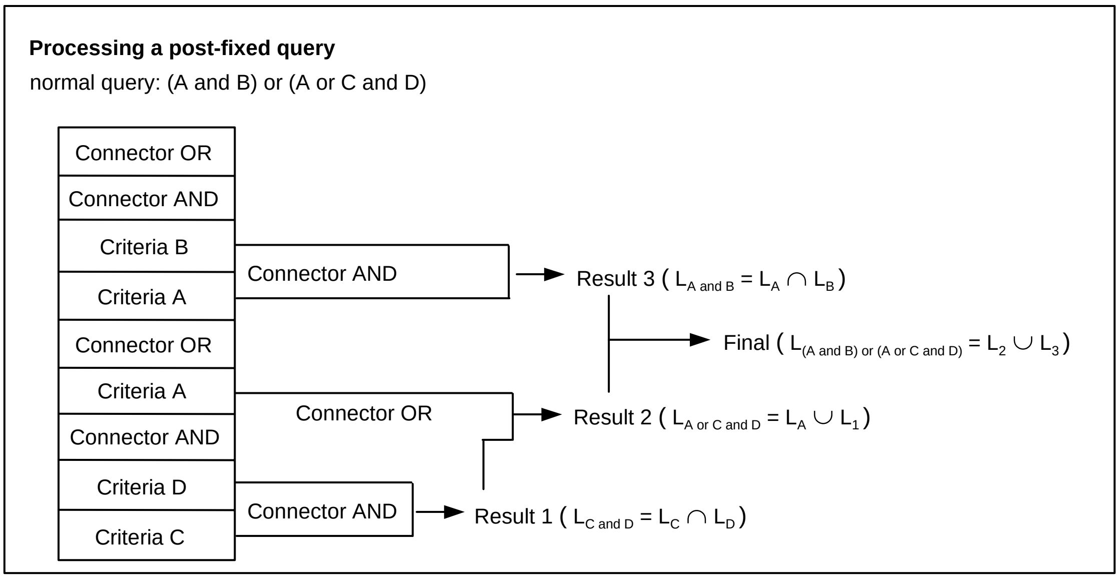
"Example of defining a query" on page 97 details the steps necessary to construct and execute a query.

Executing a query

- Define the query. For available methods, see "Define a query" on page 98.
- Execute a readFirst to return the list of documents that match the query criteria. For more information, see "Navigate in the list" on page 100.
- Once a record has been accessed, you can get information on the document. For more information, see "Access a record" on page 101.

Post-fixed processing

The following shows the processing of a generic query.



Processing a post-fixed query
normal query: (A and B) or (A or C and D)
Connector OR
Connector AND
Criteria B
Criteria A
Connector OR
Criteria A
Connector AND
Criteria D
Criteria C
Connector AND
Result 3 ($L_{A\ and\ B} = L_A \cap L_B$)
Final ($L_{(A\ and\ B)\ or\ (A\ or\ C\ and\ D)} = L_2 \cup L_3$)
Connector OR
Result 2 ($L_{A\ or\ C\ and\ D} = L_A \cup L_1$)
Connector AND
Result 1 ($L_{C\ and\ D} = L_C \cap L_D$)

How the stack is processed

Step 1:

When the Boolean connector AND is found, it is applied to Criteria D and to Criteria C, found below it in the stack, providing the Result 1.

Step 2:

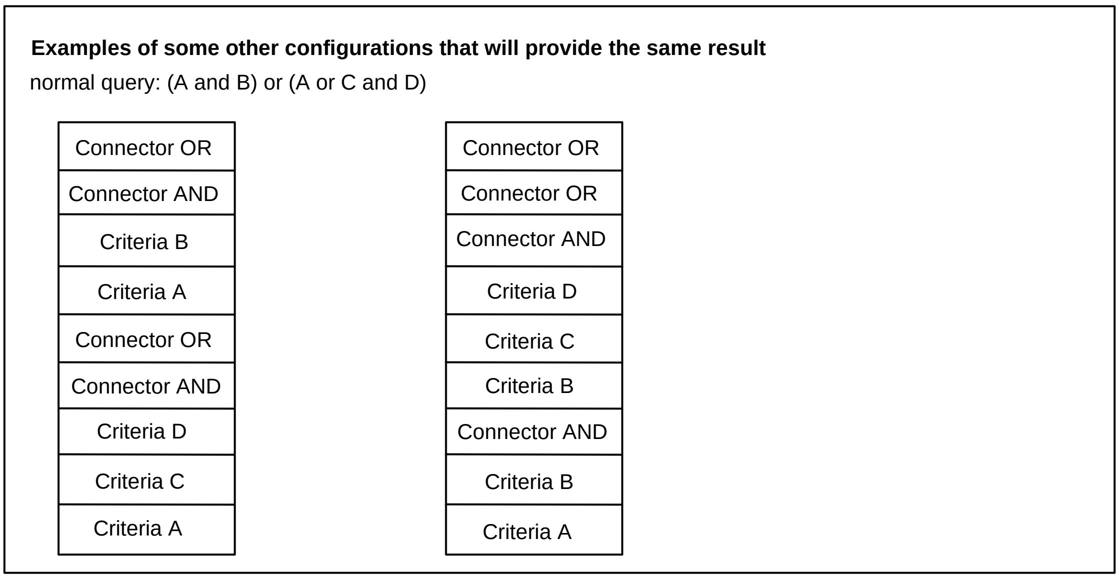
When the Boolean connector OR is found, it is applied to Criteria A and (because there is not another set of criteria in the stack) to Result 1, providing the Result 2.

Step 3:

When the Boolean connector AND is found, it is applied to Criteria B and to Criteria A, the two criteria found below it in the stack, providing the Result 3.

Step 4:

When the last Boolean connector OR is found, it is applied to Result 3 and Result 4 (the last two completed operations).



Example of defining a query

Assume that the database contains the following index field descriptors (indexes):

- account (index of account numbers for clients)
- due_amt (index of the amount due for each client)
- name (index of client names)

A search is performed on the three indexes to find documents that match the following criteria:

(account containing the string 58) OR (due_amt > 100 AND name <> ADAMS)

Thus, there are 3 sets of criteria (shown in the table below):

Set of	FieldName	operator	value
Criteria 1	Account	FLD_OP_CRITERIA_REGULAR	.*58.*
Criteria 2	due_amt	FLD_OP_CRITERIA_GREATER	100
Criteria 3	Name	FLD_OP_CRITERIA_NOT_EQUAL	ADAMS

Steps to send the query to the server

myQuery is a valid FLDQueryList object.

1 When executed, the following method sends the first set of criteria.

```
myQuery.addQueryCriteria("account",FLDQueryList.FLD_OP_CRITERIA_REGULAR,".*58.*");
```

2 When executed, the following method sends the second set of criteria.

```
myQuery.addQueryCriteria("due_amt",FLDQueryList.FLD_OP_CRITERIA_GREATER,"100");
```

3 When executed, the following method sends the last set of criteria.

```
myQuery.addQueryCriteria("name",FLDQueryList.FLD_OP_CRITERIA_NOT_EQUAL,"ADAMS");
```

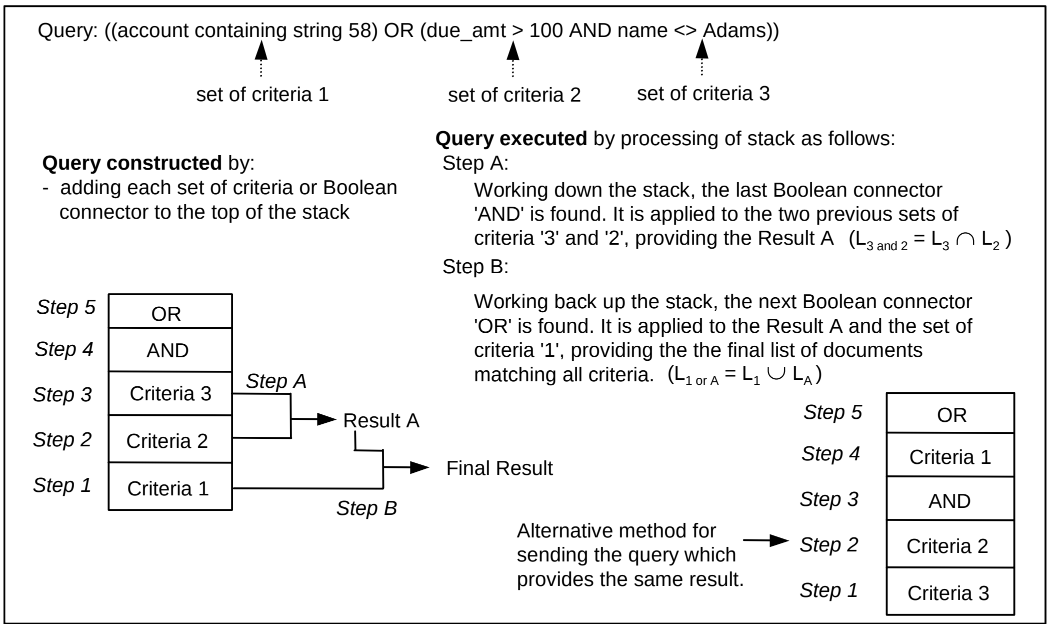
4 When executed, the following sends the Boolean connector that links the 2nd and 3rd sets of criteria.

```
myQuery.addQueryOperator(FLDQueryList.FLD_OP_QUERY_AND);
```

5 When executed, the following method sends a Boolean connector that links the result of the 1st and 2nd sets of criteria and the first set of criteria.

```
myQuery.addQueryOperator(FLDQueryList.FLD_OP_QUERY_OR);
```

The query has now been constructed. When a readFirst method is executed, the search is performed, based on the query, and a list of matching documents is accessible.


Query: ((account containing string 58) OR (due_amt > 100 AND name <> Adams))
set of criteria 1
set of criteria 2
set of criteria 3
Query constructed by:
- adding each set of criteria or Boolean connector to the top of the stack
Query executed by processing of stack as follows:
Step A:
Working down the stack, the last Boolean connector 'AND' is found. It is applied to the two previous sets of criteria '3' and '2', providing the Result A ($L_{3 \text{ and } 2} = L_3 \cap L_2$)
Step B:
Working back up the stack, the next Boolean connector 'OR' is found. It is applied to the Result A and the set of criteria '1', providing the the final list of documents matching all criteria. ($L_{1 \text{ or } A} = L_1 \cup L_A$)
Step 5
OR
Step 4
AND
Step 3
Criteria 3
Step 2
Criteria 2
Step 1
Criteria 1
Step A
Result A
Final Result
Step B
Alternative method for sending the query which provides the same result.
Step 5
OR
Step 4
Criteria 1
Step 3
AND
Step 2
Criteria 2
Step 1
Criteria 3

Each time the methods addQueryCriteria or addQueryOperator is executed, the set of criteria or Boolean connector is added to the stack.

Define a query

The following methods define the query. Minimally, the addQueryCriteria method must be executed if only one set of criteria is to be used. If more than one set of criteria is defined, the addQueryOperator must also be executed to define the Boolean connector. Once the criteria are defined, use the additional methods to further restrict the query. Note: To be taken into account, the enable and set methods that define other query restrictions and criteria must be executed before executing a readFirst method.

addQueryCriteria

The execution of this method defines and sends one set of criteria to the server.

Method: addQueryCriteria(String fieldName, byte[] operator, String value)

Parameters:

- "fieldName" is the name of the index field descriptor whose entries will be searched. To obtain a list of index field descriptors, see "Retrieve a list of index field descriptors" on page 92.
- "operator" is the comparison operator to use when matching the index entries to the specified value. Permissible values (information in parenthesis):
  - FLD_OP_CRITERIA_EQUAL (=)
  - FLD_OP_CRITERIA_NOT_EQUAL (<>)
  - FLD_OP_CRITERIA_GREATER (>)
  - FLD_OP_CRITERIA_GREATER_OR_EQUAL (>=)
  - FLD_OP_CRITERIA_LOWER (<)
  - FLD_OP_CRITERIA_LOWER_OR_EQUAL (<=)
  - FLD_OP_CRITERIA_REGULAR (regular expression, i.e. standard expression used in Perl)
- "value" is the string that is matched. Permissible values: any value or regular expression.

addQueryOperator

The execution of this method sends a Boolean connector to the server.

Method: addQueryOperator(byte[] operator)

Parameter: "operator" is the Boolean connector to use when linking sets of criteria. Permissible values (information in parenthesis):

- FLD_OP_QUERY_AND (AND Boolean connector)
- FLD_OP_QUERY_OR (OR Boolean connector)

addToFilterList

Permits the definition of one or more filters for limiting the list of documents authorized for retrieval. This method is only available if the method "enableFilterList" documented on page 99 has been executed. For general information, see "Query results returned by the server" on page 95.

Method: addToFilterList(String filterName)

Parameter: "filterName" is name of an F/D filter authorized for the logged-on user.

enableDocumentDetails

An instance of the FLDDocument interface is used to return information on the document. Normally, only a subset of the methods are available. This method allows you make more methods available. For the information returned, see "General document information" on page 60.

Method: enableDocumentDetails(boolean docDetails)

Parameter: "docDetails" set to:

- True, permits retrieval of additional document details.
- False, this information is not available.

enableDocumentID

An instance of the FLDDocument interface is used to return information on the document. Normally, only a subset of the methods are available. This method allows you make more methods available. For the information returned, see "General document information" on page 60.

Method: enableDocumentID(boolean docId)

Parameter: "docId" set to:

- True, permits retrieval of the documentID and the rawDocument ID.
- False, this information is not available.

enableFilterList

This method must be executed to activate functionality to change the filters that are used to determine what documents will be returned by a query. For more information, see "Query results returned by the server" on page 95.

Method: enableFilterList(boolean state)

Parameter: "state" when set to True:

- Initially sets the list of filters used to return query results to all filters authorized for the logged-on user.
- Activates the method "addToFilterList" documented on page 98.
- Activates the method "getFilterName" documented on page 101.

When set to False (the default), the above methods are not available and the server uses the active filter name to determine which documents to include in the query result list.

enableIndexInfo

This method determines whether or not information on the index values will be available when information is retrieved using the 'next' navigation method.

Method: enableIndexInfo(boolean indexInfo)

Parameter: "indexInfo" set to:

- True, when the getIndexes method available from an instance of the FLDQueryListEntry interface (page 101) is executed, returns information on the index field names and index values for the document.
- False, the getIndexes method returns nothing.

enableRunInfo

This method determines whether or not information on a run will be available when information is retrieved using the 'next' navigation method.

Method: enableRunInfo(boolean runInfo)

Parameter: "runInfo" set to:

- True, the methods getRunDate and getRunNumber (page 63) can be executed to return information.
- False, these methods return nothing.

resetQuery

Resets the query. All information sent by the previous methods is ignored.

Method: resetQuery()

setDBSearchConstraint

Used to optimize the search if more than one index is searched and the size of at least one of the indexes is significant.

Method: setDBSearchConstraint(int dbSearchNumber)

Parameter: "dbSearchNumber" is the number to limit the intermediary results.

setFolderTypeConstraint

Restricts the query results to documents that have a folder type that matches the folderType criteria.

Method: setFolderTypeConstraint(int folderType)

Parameter: "folderType" is the folder type criteria. For folder type information, see "Retrieve a list of folder types" on page 34.

setFromDateConstraint

Restricts the query results to documents created on or after the data specified. For the setFromDateConstraint and setToDateConstraint methods: if one is used, they must both be used.

Method: setFromDateConstraint(String fromDate)

Parameter: "fromDate" is a date. Format: YYYY/MM/DD.

getFromDateConstraint

Method: getFromDateConstraint()

Returns: String. The value set by the setFromDateConstraint.

setGlobalQuery

Method: setGlobalQuery(boolean global) **

Parameter: "global" set to True, returns all documents and ignores filter criteria. Set to False, returns documents using and filter criteria that has been set. The default is to use the filter criteria if it has been set in the object constructor.

setMaximumNumberDocumentsConstraint

Restricts the number of documents returned when the query is executed.

Method: setMaximumNumberDocumentsConstraint(int maxNumber)

Parameter: "maxNumber" is the maximum number of documents to return in the document list.

setRecordsByBlockConstraint

Restricts the number of documents returned in a block when the query is executed.

Method: setRecordsByBlockConstraint(int recordsByBlock)

Parameter: "recordsByBlock" is the number of documents to return per block.

setToDateConstraint

Restricts the query results to documents created on or before the data specified. See setFromDateConstraint method above.

Method: setToDateConstraint(String toDate)

Parameter: "toDate" is a date. Format: YYYY/MM/DD.

getToDateConstraint

Method: getToDateConstraint()

Returns: String. The value set by the setToDateConstraint.

Navigate in the list

The following methods navigate forward in a list of query results. To verify the existence of more records, use the hasNext method.

Initially, a readFirst method must be executed. Once a 'read' method has been executed, the next method retrieves the information for that entry and moves to the next record.

Note that the following methods are for internal use: isAtBottom, isAtTop.

hasNext

Method: hasNext()

Returns: boolean. True, if the next method can be used to retrieve information on an entry. False, otherwise.

next

Populates an instance of the **FLDQueryListEntry** interface with information contained in the record and then moves to the next record. See "Access a record" below.

Method: next() **

readFirst

Positions at the first record in the list.

Method: readFirst() **

readNext

Positions at the next record in the list. Assumes a readFirst has been executed.

Method: readNext() **

Get information on the query results

When a 'read' method is executed, the following methods return information.

getMatchingIndexNames

Method: getTotalDocumentsRetrieved()

Returns: java.util.ArrayList. The names of the indexes that contain values matching the query criteria.

getNumberOfDocuments

Method: getNumberOfDocuments()

Returns: int. The total number of documents matching the query criteria. If this information is not available, returns the value -1.

getTotalDocumentsRetrieved

Method: getTotalDocumentsRetrieved()

Returns: int. The total number of documents that have already been accessed by executing the next method.

Access a record

When the 'next' method is executed, information can be retrieved on the document itself and on the index values associated with the document. The 'next' method populates an instance of the **FLDQueryListEntry** interface with information contained in the record.

The method unique to this interface is documented below. For additional methods available once there is a valid instance of this interface, see "General document information" on page 60.

getFilterName

This method is only available if the method "enableFilterList" documented on page 99 has been executed.

Method: getFilterName()

Returns: String. The name of the filter used to authorize the inclusion of the document in the query result list. Note that for a query that has used multiple filter names, if the filters have overlapping criteria, the server will return the name of the first filter (alphabetic order) that is used to include the document.

getIndexes

This method retrieves index information for each index that was used in the initial query that created the list of documents and that has entry values associated with the document in the list of results.

Notes:

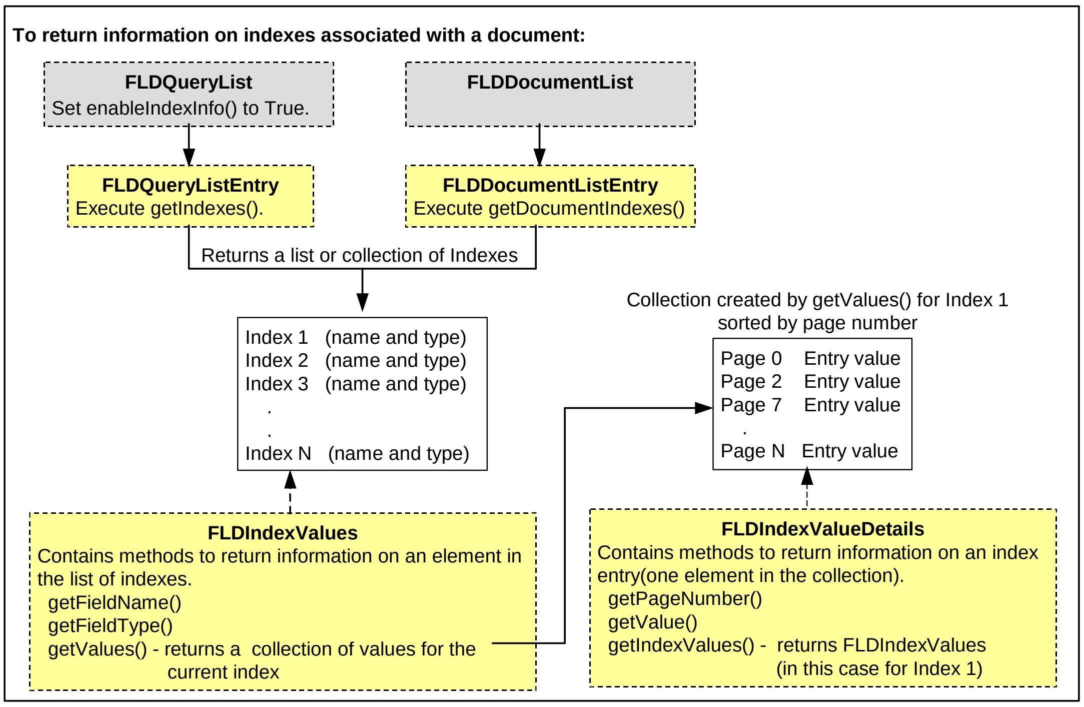
- The method is valid only if connected to an RSD Folders server running on a Unix or Windows platform where ABI functionality is implemented.
- This information is only retrieved if the method 'enableIndexInfo' has been executed set to True. See page 99.

Method: getIndexes()

Returns: java.util.Collection. The methods to return information on an entry in the list are contained in the FLDIndexValues interface. For more information, see "Get information on the indexes" on page 102.

Get information on the indexes

The image below provides an overview of the steps required to return information on index entries.


To return information on indexes associated with a document:
FLDQueryList
Set enableIndexInfo() to True.
FLDDocumentList
FLDQueryListEntry
Execute getIndexes().
FLDDocumentListEntry
Execute getDocumentIndexes()
Returns a list or collection of Indexes
Index 1 (name and type)
Index 2 (name and type)
Index 3 (name and type)
Index N (name and type)
Collection created by getValues() for Index 1 sorted by page number
Page 0 Entry value
Page 2 Entry value
Page 7 Entry value
Page N Entry value
FLDIndexValues
Contains methods to return information on an element in the list of indexes.
getFieldName()
getFieldType()
getValues() - returns a collection of values for the current index
FLDIndexValueDetails
Contains methods to return information on an index entry(one element in the collection).
getPageNumber()
getValue()
getIndexValues() - returns FLDIndexValues (in this case for Index 1)

Summary of the indexes that can be returned

When the getIndexes() method or getDocumentIndexes is executed, a list of indexes is returned.

The indexes returned depend on what method was executed.

Method	Interface	Names returned in list
getIndexes	FLDQueryListEntry	Indexes used to define initial query
getDocumentIndexes	FLDDocument	All indexes that contain entries associated with the document.
getDocumentIndexes(myList)	FLDDocument	Indexes specified in myList parameter

To return information on an entry in the list of indexes

Navigate in the list or collection of indexes to position at an element. Then use the methods in the **FLDIndexValues** interface to return information on the element.

- **getFieldName**

  Method: getFieldName()

  Returns: String. The name of the index.

- **getFieldType**

  Method: getFieldType()

  String. The format defined for the index. For more information, see "Index field types" on page 94.

- **getValues**

  Method: getValues()

  Returns: Collection of FLDIndexValueDetails. For more information, see "To return information on index entry values" on page 103.

To return information on index entry values

When the getValues() method is executed, a collection of index entry values is returned. The collection includes all entries associated with the specific index (for that document). It is sorted by page number.

Navigate in the collection to position at an element. Then use the methods in the **FLDIndexValueDetails** interface to return information on each index entry.

- **getPageNumber**

  Method: getPageNumber()

  Returns: int. The page number where the value is found in the document. A page number of 0 indicates that the index entry is modifiable (instead of indicating a physical location). For more information, see "Add, delete or change index entries" on page 103.
- **getValue**

  Method: getValue()

  Returns: java.Lang.Object. The index entry value. The format of the object depends on the field type, which is returned by the getFieldType method discussed previously.
- **getIndexValues**

  Method: getIndexValues()

  Returns: FLDIndexValues. The current index information.

Add, delete or change index entries

The **FLDIndexFieldUpdate** interface permits the addition, deletion or modification of an index entry value.

General information

An instance of this interface is implemented when the newIndexFieldUpdate method is executed from an instance of the FLDConnection object (page 28) or the FLDDocument interface (page 65).

The logged user name must authorized to perform the functionality. Depending on the function executed, one of the following methods must return true: isIndexAddAuthorized, isIndexDeleteAuthorized, isIndexModifyAuthorized. For more information on authorizations, see "Get information on user authorizations" on page 27.

An index entry contains the following information:

- Index entry value.
- Date that the index entry is added or modified.
- Folder type and doc key. The index value is associated with the document specified by these parameters.
- Page number. Index values that are added manually, by a user or an administrator are assigned a page number value of 0. Only index values with a page number 0 can be modified.

To add, delete or update an index value:

Note: All methods mentioned are documented following.

1 Execute the newIndexFieldUpdate method. It creates an instance of the FLDIndexFieldUpdate interface. If the method is executed from:
   - An instance of the FLDConnection object, then the setDocumentInfo method documented following must be executed.
   - An instance of the FLDDocument interface, make sure that the instance is pointing to the document associated with the index entry to be modified or deleted or the document where the index entry is to be added.

2 Execute one of the following methods: createAddAction, createDeleteAction, createUpdateAction.

3 For add and delete action:
   - Execute the setField method to indicate the Index Field Descriptor.
   - Execute the setValue method to indicate the index entry value.

   For an index entry update action:
   - Execute the setUpdate method to indicate the value to update.

4 For all actions, apply the execute method.

Initial methods to add, modify or delete an index entry value

The following methods, available in the **FLDIndexFieldUpdate** interface, create valid instances to enable the addition, deletion or update of an index entry value.

createAddAction

Permits the addition of an index entry value.

Method: createAddAction()

Returns: An instance of the FLDIndexFieldUpdate.AddAction interface. The interface inherits methods from the FLDIndexFieldUpdate.Action interface documented on page 105.

createDeleteAction

Permits the deletion of an index entry value.

Method: createDeleteAction()

Returns: An instance of the FLDIndexFieldUpdate.DeleteAction interface. The interface inherits methods from the FLDIndexFieldUpdate.Action interface documented on page 105.

createUpdateAction

Permits the modification of an index entry value.

Method: createUpdateAction()

Returns: An instance of the FLDIndexFieldUpdate.UpdateAction interface. The interface inherits methods from the FLDIndexFieldUpdate.UpdateAction interface and the FLDIndexFieldUpdate.Action interface documented following.

Method to update an index entry value

In addition to the setUpdate method, the **FLDIndexFieldUpdate.UpdateAction** interface inherits methods from the FLDIndexFieldUpdate.Action interface.

setUpdate

This method specifies the index where the entry value exists and indicates both the entry value to be modified and the new entry value. There are two implementations of this method.

Method: setUpdate(FLDIndexFieldListEntry field, java.lang.Object oldValue, java.lang.Object newValue)

Method: setUpdate(String fieldType, String fieldName, java.lang.Object oldValue, java.lang.Object newValue)

Parameters:

- For field, fieldType and fieldName see the setField method below.
- "oldValue" indicates the value of the index entry to be modified.
- "newValue" indicates the updated value of the index entry.

Note: For "oldValue" and "newValue", the format for the java.lang.Object must be in compliance with the format defined for the index field.

Methods to add, modify or update an index entry value

These methods contained in the **FLDIndexFieldUpdate.Action** interface are inherited by instances of the following interfaces: FLDIndexFieldUpdate.AddAction, FLDIndexFieldUpdate.DeleteAction, FLDIndexFieldUpdate.UpdateAction.

execute

Executes an action: add, delete or update an index entry value (depending on the interface implemented).

Method: execute() **

Returns the sequence number if the request is done via a FLDIndexUpdateTransaction.

setDocumentInfo

This method is required if an instance of the FLDIndexFieldUpdate interface is created from an instance of the FLDConnection object. Otherwise, it need not be executed.

Method: setDocumentInfo(int folderType, byte[] docKey)

Parameters: The following parameters indicate the document where the index value exists (if it is to be deleted or modified) or the document with which an added index value should be associated.

- "docKey" is the document key.
- "folderType" is the folder type.

setField

This method sets the field (index) where the entry value exists, if it is to be deleted, or where a new entry value should be added. There are two implementations of this method.

Method: setField(FLDIndexFieldListEntry field)

Method: setField(String fieldType, String fieldName)

Parameters:

- "field" is an instance of the FLDIndexFieldListEntry interface indicating the index where the value will be added or deleted.
- "fieldType" indicates the format for the index where the entry value will be added or deleted. It must conform with the definition on the RSD Folders server accessed. Permissible value: a single character. For more information, see "Index field types" on page 94.
- "fieldName" is the name of the index where the entry value will be added or deleted.

setValue

This method sets the value for the index entry that should be added or deleted.

Method: setValue(java.lang.Object entryValue)

Parameter: "entryValue" is the value for the index entry to be added or deleted. The format must be in compliance with the format defined for the index field.

Index Update transaction

The **FLDIndexUpdateTransaction** object initializes and terminates a transaction between the client application and an RSD Folders server. If the object is initialized all update/create/delete requests to the ABI are executed only when the FLDIndexUpdateTransaction object is closed. This transaction improves performance on the server when updating multiple ABI values.

Constructor: FLDIndexUpdateTransaction(FLDConnection connectInfo, java.lang.String Desc)

Parameter: a valid FLDConnection and a description to be passed to the server for tracking

getServerTaskID

This method gives the server ID when the transaction has been opened.

Method: getServerTaskID()**

Returns: String. The server ID.

getTransactionID

Gives the transaction ID when the transaction has been closed.

Method: **getTransactionID**()

Returns: int. The transaction ID when completed, otherwise zero.

getNumberOfRequest

Gives the number of requests processed for the transaction when the transaction has been closed.

Method: getNumberOfRequest()

Returns: int. The number of request processed.

getNumberOfRequestError

Gives the number of request, in error for the transaction when the transaction has been closed.

Method: getNumberOfRequestError()

Returns: int. The number of request, in error.

getStatusRequest

Gives for a sequence the return code of the request.

Method: getStatusRequest(int sequence)

Returns: an FLDException if an error occured during the request, otherwise is null.

close

Notifies the server to stop processing for this transaction.

Method: close(String Desc)**

Parameters: description to be passed to the server for tracking.

Document data retrieval objects

Section overview

Below is a list of the objects used to retrieve document data.

Easiest method for transferring a document

The FLDDocumentTransfer object can be used to transfer a document in any format.

It differs from the other objects that perform document transfer because it transfers the complete document at one time. Other objects transfer document information a single line or block at a time.

When transferring AFP documents, with this object, the resources are also transferred. With other transfer objects, the resources must be transferred separately.

For text documents, the FLDTextDocumentTransfer object should be used only if the transfer should be restricted to a specific portion of the document. To transfer the whole document, use the FLDDocumentTransfer object.

Transferring AFP resources

To avoid transferring the same resources each time a document (using them) is transferred they should be saved locally. To activate this functionality, see "setResourcesPath" on page 23.

Objects covered in this section:

- "Transfer any document" on page 108 (FLDDocumentTransfer)
- "Restrict information in a text document" on page 112 (FLDTextDocumentFind)
- "Transfer a text document" on page 109 (FLDTextDocumentTransfer)
- "Transfer a binary document" on page 114 (FLDBinaryDocumentTransfer)
- "Transfer an AFP document" on page 115 (FLDAFPDocumentTransfer)
- "Retrieve a document resource list" on page 116 (FLDDocumentResourceList)
- "Transfer document resources" on page 118 (FLDResourceTransfer)

For a diagram showing the objects, see "Diagram of objects" on page 15.

Transfer any document

The **FLDDocumentTransfer** object transfers any logical document in any format to one file or output stream. For simple transfers, it is the preferred transfer object. For more information, see "Easiest method for transferring a document" on page 107.

Constructor: FLDDocumentTransfer(FLDDocumentList documentList) **

Parameter: "documentList" is a valid instance of the FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDDocumentTransfer(FLDQueryList queryList) **

Parameter: "queryList" is a valid instance of the FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDDocumentTransfer(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Notes
If the document has resources associated with it, the resources are also transferred. To avoid transferring the same resources each time a document (using them) is transferred they should be saved locally. To activate this functionality, see "setResourcesPath" on page 23.

If a document is not on direct access media, then an error (either 26006 or 28006) is returned. To successfully transfer the document, re-execute the transfer method to confirm the transfer (the second time there will be no error).

The following methods are for internal use: getFromOffset, getToOffset, setFromOffset, setToOffest.

To transfer a document

Transfer a document
All methods return void.

Method: transfer(FLDDocument document, OutputStream outputStream) **

Method: transfer(FLDDocument document, String pathName) **

Parameters:

- "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.
- "outputStream" is the stream used to receive the data.
- "pathName" is the full path name of the file to create, including the file name.
  - If the extension (for the file name) is included in "pathName", it is used.
  - If the extension is omitted, then when the file is created, the file extension is automatically added to the file name. To retrieve the full name for file, after it is created, use the "getPathName" method documented below. Some examples:

document extension	file name	file created
100	docName	docName.txt
300	docName	docName.afp
doc	docName	docName.doc

Get information on the transferred data

The first method, getPathName, is only used if the method for transfer included a pathName.

getPathName

Method: getPathName()

Returns: String. The name of the created file. The returned String includes path directory, file name and file extension.

Transfer a text document

The **FLDTextDocumentTransfer** object transfers all or part of text document one line (record) at a time. To transfer the complete document, use the FLDDocumentTransfer object. For more information, see "Transfer any document" on page 108.

Constructor: FLDTextDocumentTransfer(FLDDocumentList documentList) **

Parameter: "documentList" is a valid instance of the FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDTextDocumentTransfer(FLDQueryList queryList) **

Parameter: "queryList" is a valid instance of the FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDTextDocumentTransfer(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Notes

If a document is not on direct access media, then the error 26006 is returned. To successfully transfer the document, re-execute the transfer method to confirm the transfer (the second time there will be no error).

Note that the following method is for internal use: setTransferObjective, first, next, has next, isAtBottom, readNextLine, readNextPage, getTotalLinesRead

Restrict the information returned

These set methods restrict the information returned using the "read" methods for the object.

resetFindRequest

This method resets any settings made with the setFindRequest method.

Method: resetFindRequest()

setBeginColumn

Method: setBeginColumn(short beginCol)

Parameter: "beginCol" indicates to the server, the column on the line to use as the starting point for returning data.

setEndColumn

Method: setEndColumn(short endCol)

Parameter: "endCol" indicates to the server, the column on the line to use as the ending point for returning data.

setFindRequest

Method: setFindRequest (FLDTextDocumentFind docFind)

Parameter: "docFind" is a valid instance of the FLDTextDocumentFind object. Limits the text to transfer. For more information, see "Restrict information in a text document" on page 112.

setMaxLine

Method: setMaxLine(short maxLine)

Parameter: "maxLine" indicates to the server, the maximum number of lines to return when a "read" method is used. Note that for an RSD Folders Server running on a z/OS platform, this number must be within the default range set on the server.

Navigate in the document

The following methods access one or more lines or pages in the document.

The methods return boolean: True, a line is available; False, otherwise.

Parameters for all navigation methods:

- "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.
- "lineNumber" is a line number in the document.
- "pageNumber" is a page number in the document.

hasNext

Method: hasNext()

Returns: boolean. True, if the following navigation methods will return a line of text. False, otherwise.

readFirst

Accesses the first line in the specified document.

Method: readFirst(FLDDocument document) **

readLast

Accesses the last line in the specified document.

Method: readLast(FLDDocument document) **

readNext

Accesses the next line in the document. Assumes a readFirst, readNext, readNextPage, readNextLine, readPositionPage or readPositionLine has been executed.

Method: readNext() **

readNextLine

Accesses the line after the line specified by the lineNumber parameter.

Method: readNextLine(FLDDocument document, int LineNumber) **

readNextPage

Accesses the first line on the page after the specified page.

Method: readNextPage(FLDDocument document, int pageNumber) **

readPositionLine

Accesses the line specified by the lineNumber parameter.

Method: readPositionLine(FLDDocument document, int lineNumber) **

readPositionPage

Accesses the first line for the specified page.

Method: readPositionPage(FLDDocument document, int pageNumber) **

Access information

When any 'read' method is executed, the following methods return information.

getFLDTextLine

Method: getFLDTextLine()

Returns: An instance of the FLDTextLine interface with information contained in the record. See 'Retrieve the data returned on the line' below.

Retrieve the data returned on the line

The following methods are contained in the **FLDTextLine** interface

getLineNumber

Method: getLineNumber()

Returns: int. The line number.

getPageNumber

Method: getPageNumber()

Returns: int. The page number for the line.

getPrintCode

Method: getPrintCode()

Returns: char. The print code.-

getTextLine

Method: getTextLine()

Returns: String. The document line data.

isArgumentFound

Method: isArgumentFound()

Returns: boolean. True, the string specified in the find criteria is contained in the line. False, otherwise.

isNewPage

Method: isNewPage()

Returns: boolean. The status of the page. True indicates a new page. False otherwise.

Restrict information in a text document

The **FLDTextDocumentFind** object sets, for a text document, the scope of the search and the search criteria. An instance of this object can be referenced by the FLDTextDocumentTransfer object to return only lines of text that match the definition set for the instance.

Constructor: FLDTextDocumentFind( )

Set methods

setBackward

Method: setBackward(boolean flag)

Parameter: "flag" is set: True, search from the start line toward the beginning of the document; False, search forward. Default: False.

setBeginColumn

Method: setBeginColumn(short beginCol)

Parameter: "beginCol" indicates to the server, the column on the line to use as the starting point for returning data.

setEndColumn

Method: setEndColumn(short endCol)

Parameter: "endCol" indicates to the server, the column on the line to use as the ending point for returning data.

setFindArgument

Method: setFindArgument(String findArgument)

Parameter: "findArgument" indicates the text to search for.

setLineRange

Method: setLineRange(long lineRange)

Parameter: "range" is the maximum number of document lines that should be searched.

setMatchCase

Method: setMatchCase(boolean flag)

Parameter: "flag" if set to True, matches the case; False ignores the character case.

setOperator

Method: setOperator(int type)

Parameter: "type" indicates how to match document data with the string. Permissible values: DIFFERENT, EQUAL, GREATER, LESS.

setStartLine

Method: setStartLine(long startLine)

Parameter: "startLine" is the number of the first document line to search.

Get methods

getBeginColumn

Method: getBeginColumn()

Returns: short. The start column restriction as defined in the setBeginColumn method.

getEndColumn

Method: getEndColumn()

Returns: short. The end column restriction as defined in the setEndColumn method.

getFindArgument

Method: getFindArgument()

Returns: String. The find criteria settings, as defined in the setFindArgument method.

getLineRange

Method: getLineRange()

Returns: long. The number of document lines to limit the range to search, as defined using the setLineRange method.

getStartLine

Method: getStartLine()

Returns: long. The document line number where the search should begin, as defined using the setStartLine method.

isBackward

Method: isBackward()

Returns: boolean. True, the search is made from the start line toward the beginning of the document. False, the search is made from the start line toward the end of the document.

isDifferent

Method: isDifferent()

Returns: boolean. True, finds data that is not equal to the string. False, another method of matching data is set.

isEqual

Method: isEqual()

Returns: boolean. True, finds data that matches the string. False, another method of matching data is set.

isGreater

Method: isGreater()

Returns: boolean. True, finds data that is greater than the string. False, another method of matching data is set.

isLess

Method: isLess()

Returns: boolean. True, finds data that is less than the string. False, another method of matching data is set.

isMatchCase

Method: isMatchCase()

Returns: boolean. True, finds data that has same case as string. False, ignores the character case when finding data.

Transfer a binary document

The **FLDBinaryDocumentTransfer** object transfers a binary document one data block at a time. To transfer the complete document, use the FLDDocumentTransfer object explained on page 108.

Constructor: FLDBinaryDocumentTransfer(FLDDocumentList documentList) **

Parameter: "documentList" is a valid FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDBinaryDocumentTransfer(FLDQueryList queryList) **

Parameter: "queryList" is a valid FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDBinaryDocumentTransfer(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Note

If a document is not on direct access media, then the error 28006 is returned. To successfully transfer the document, re-execute the transfer method to confirm the transfer (the second time there will be no error).

Access a data block in the document

The methods return boolean: True, a block is available; False, otherwise.

Parameters for all navigation methods:

- "docKey" is the document key.
- "extDocKey" is the extended document key.
- "folderName" is a valid folder name within the folder type, maximum 32 characters.
- "folderType" must fall within the range of folder types defined in the active filter and valid for the logged-on user. For folder type information, see "Retrieve a list of folder types" on page 34 and "Set a filter" on page 38.
- "offset" is the data block information obtained by using the getRestartOffset (described following).

For parameters docKey, extDocKey, folderName and folderType, valid values can be returned using methods described in the topic "General document information" on page 60.

readFirst

Accesses the first data block in the document. The first method requires the use of the document list constructor and the second method requires the use of the query constructor.

Method: readFirst(byte[] docKey, short extDocKey) **

Method: readFirst(byte[] docKey, short extDocKey, String folderName, int folderType) **

readNext

Accesses the next data block in the document. Assumes a readFirst or readPosition has been executed.

Method: readNext() **

readPosition

Accesses the specified data block. This method is used to restart a transfer. The first method requires a document list constructor and the second a query constructor.

Method: readPosition(byte[] docKey, short extDocKey, int offset) **

Method: readPosition(byte[] docKey, short extDocKey, int offset, String folderName, int folderType) **

Get the information on the data block

When the method readFirst, readNext or readPosition is performed, the following methods return information contained in the data block accessed.

getBinaryData

Method: getBinaryData()

Returns: byte[]. The data block.

getDataSize

Method: getDataSize()

Returns: int. The size of the data block. The maximum size is 32 kilobytes.

getRestartOffset

Method: getRestartOffset()

Returns: int. The offset for the data block. Needed to restart a transfer positioned at a specific data block.

Transfer an AFP document

The **FLDAFPDocumentTransfer** object transfers an AFP document one data block at a time. The resources must be transferred separately. To transfer the complete AFP document and its resources, use the FLDDocumentTransfer object. For more information, see "Transfer any document" on page 108.

Constructor: FLDAFPDocumentTransfer(FLDDocumentList documentList) **

Parameter: "documentList" is a valid FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDAFPDocumentTransfer(FLDQueryList queryList) **

Parameter: "queryList" is a valid FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDAFPDocumentTransfer(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Note

If a document is not on direct access media, then the error 26006 is returned. To successfully transfer the document, re-execute the transfer method to confirm the transfer (the second time there will be no error).

Access a data block in the document

The methods return boolean: True, a line is available; False, otherwise.

Parameters for all navigation methods:

- "document" is a valid instance of the FLDDocument interface. For more information, see "General document information" on page 60.
- "lineNumber" is a line number in the document.
- "pageNumber" is a page number in the document.

readFirst

Accesses the first line in the document.

Method: readFirst(FLDDocument document) **

readNext

Accesses the next line in the document. Assumes a readFirst, readPositionPage or readPositionLine has been executed.

Method: readNext() **

readPositionLine

Accesses the line specified by the lineNumber parameter.

Method: readPositionLine(FLDDocument document, int lineNumber) **

readPositionPage

Accesses the first line for the specified page.

Method: readPositionPage(FLDDocument document, int pageNumber) **

Get the information on the data block

When the method readFirst, readPositionLine, readPositionPage or readNext is performed, the following methods return information contained in the data block accessed.

getBinaryData

Method: getBinaryData()

Returns: byte[]. The data block.

getDataSize

Method: getDataSize()

Returns: int. The size of the data block. The maximum size is 32 kilobytes.

Retrieve a document resource list

The **FLDDocumentResourceList** object retrieves a list of resources for the specified document.

Constructor: FLDDocumentResourceList(FLDDocumentList documentList) **

Parameter: "documentList" is a valid FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.

Constructor: FLDDocumentResourceList(FLDQueryList queryList) **

Parameter: "queryList" is a valid FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Constructor: FLDDocumentResourceList(FLDDocumentByID docByID) **

Parameter: "docByID" is a valid instance of FLDDdocumentbyID object. For more information, see "Direct access to document information" on page 67.

Important - To avoid transferring the same resources each time a document (using them) is transferred they should be saved locally. To activate this functionality, see "setResourcesPath" on page 23.

Access a record in the list

The methods return boolean: True, a line is available; False, otherwise.

Parameters for all navigation methods:

- "docKey" is the document key.
- "extDocKey" is the extended document key.
- "folderName" is a valid folder name within the folder type, maximum 32 characters.
- "folderType" must fall within the range of folder types defined in the active filter and valid for the logged-on user. For folder type information, see "Retrieve a list of folder types" on page 34 and "Set a filter" on page 38.

For parameters docKey, extDocKey, folderName and folderType, valid values can be returned using methods described in the topic "General document information" on page 60.

readFirst

Accesses the first record in the list. The first method requires the use of the document list constructor and the second method requires the use of the query constructor.

Method: readFirst(byte[] docKey, short extDocKey) **

Method: readFirst(byte[] docKey, short extDocKey, String folderName, int folderType) **

readNext

Accesses the next record in the list. Assumes a readFirst has been executed.

Method: readNext() **

Get the information in the record

When the method readFirst or readNext is performed, the following methods return information contained in the record accessed. The method 'getNumberResources', returns information on the list.

getNumberResources

Method: getNumberResources()

Returns: int. The number of resources in the list.

getResourceCheckSum

Method: getResourceCheckSum()

Returns: long. The resource checksum value.

getResourceName

Method: getResourceName()

Returns: String. The resource name.

getResourceSize

Method: getResourceSize()

Returns: long. The resource size.

getResourceStatus

Method: getResourceStatus()

Returns: byte[]. The resource status. Returned values: 0 (resource offline); 1 (resource online); > 1 (an error).

getResourceType

Method: getResourceType()

Returns: char. The resource type. Returned values: F (Formdef); O (Overlay); P (PageSegment); Z (Font Definition).

getResourceVersion

Method: getResourceVersion()

Returns: long. The resource version.

Transfer document resources

The **FLDResourceTransfer** object transfers specific document resources.

Constructor: FLDResourceTransfer(FLDDocumentResourceList documentResourceList) **

Parameter: "documentResourceList" is a valid FLDDocumentResourceList object.

Access a data block of a resource

The methods return boolean: True, a block is available; False, otherwise.

Parameters for all navigation methods:

- "resName" is the resource name.
- "resType" is the resource type.
- "resVersion" is the resource version.
- "extDocKey" is the extended document key.

For the parameter extDocKey, a valid value can be returned using methods described in the topic "General document information" on page 60. For all other parameters, values can be returned using methods described in the topic "Retrieve a document resource list" on page 116.

readFirst

Accesses the first data block of a resource.

Method: readFirst(String resName, char resType, int resVersion, short extDocKey) **

readNext

Accesses the next data block of the resource. Assumes a readFirst has been executed.

Method: readNext() **

Get the information on the data block

When the method readFirst or readNext is performed, the following methods return information contained in the data block accessed.

getBinaryData

Method: getBinaryData()

Returns: byte[]. The data block.

getDataSize

Method: getDataSize()

Returns: int. The size of the data block. The maximum size is 32 kilobytes.

Extraction objects

Section overview

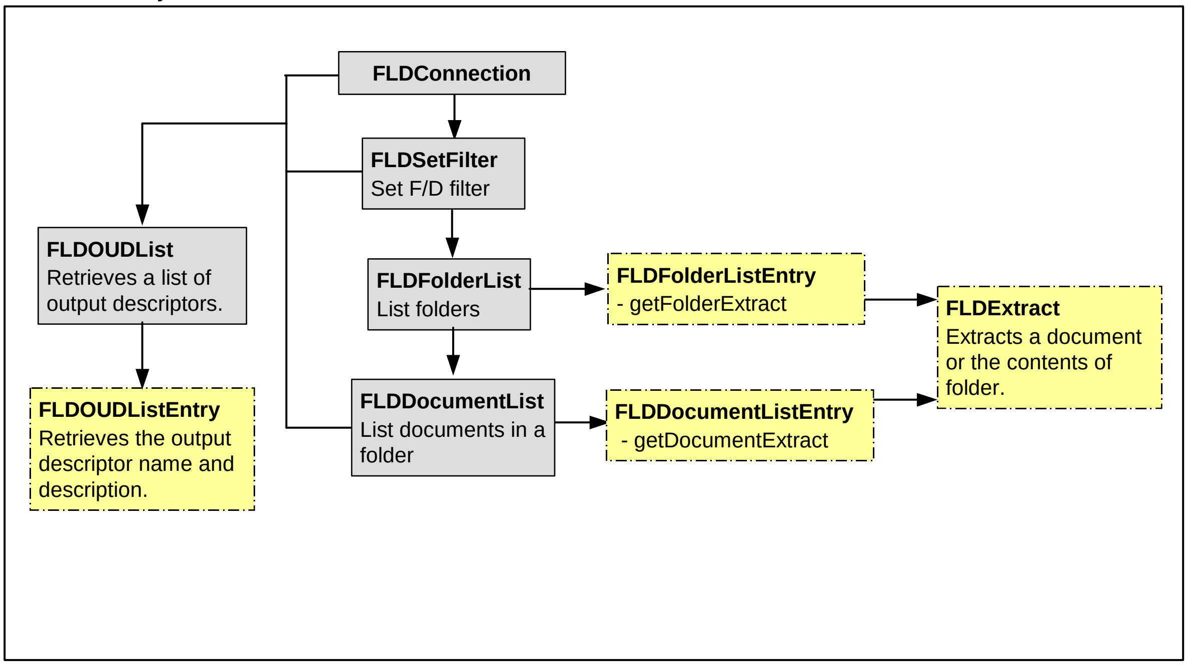
Below is a list of the objects related to extraction functionality.

Objects covered in this section

- "Extract a folder or document" on page 120. (FLDExtract)
- "Retrieve a list of output descriptors" on page 122. (FLDOUDList)

The image below is designed to show the relationship between the objects.

Extraction objects



Extract a folder or document

The **FLDExtract** interface extracts a document or the documents contained in a folder.

It is implemented by executing one of the following methods:

- getDocumentExtract documented on page 60 and page 67.
- getDocumentExtract documented on page 60 and page 67.
- "createFolderExtract" documented on page 49.

Set and execute methods

These methods are grouped, by function. The set methods return boolean: True, the method was successfully executed; False, either there is a problem with the information sent by the method or the logged-on user is not authorized to perform the action.

execute

Executes the extraction of a document or folder (of documents). The set methods must be executed before using this method.

Method: execute()

setAFPInterpreted

Method: setAFPInterpreted(boolean interpreted)

Parameter: "interpreted" if True, AFP documents are converted into text document. If False, they are not converted.

Notes:

- To execute this method, the isAFPInterpretedModifyAuthorized method must return True.
- This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.

isAFPInterpretedModifyAuthorized

Method: isAFPInterpretedModifyAuthorized()

Returns: boolean. True, if the document is in AFP format. False, Otherwise.

setCollate

Method: setCollate(boolean collate)

Parameter: "collate" if True, documents are collated. If False, they are not collated.

Note: To execute this method, the isCollateModifyAuthorized method must return True.

isCollateModifyAuthorized

Method: isCollateModifyAuthorized()

Returns: boolean. True, if the logged-on user is authorized for this action. False, otherwise.

setInternalPrintCharacteristics

Method: setInternalPrintCharacteristics(boolean ipc)

Parameter: "ipc" if True, extraction applies InternalPrintCharacteristics. If False, they are not applied.

Notes:

- To execute this method, the isInternalPrintCharacteristicsModifyAuthorized method must return True.
- This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.

isInternalPrintCharacteristicsModifyAuthorized

Method: isInternalPrintCharacteristicsModifyAuthorized()

Returns: boolean. True, if the logged-on user is authorized for this action. False, otherwise.

setOUDName

Method: setOUDName(boolean name)

Parameter: "name" is the name of a defined output descriptor.

Note: To execute this method, the isOUDNameModifyAuthorized method must return True.

isOUDNameModifyAuthorized

Method: isOUDNameModifyAuthorized()

Returns: boolean. True, if the logged-on user is authorized for this action. False, otherwise.

setOutputType

Method: setOutputType(int type)

Parameter: "type" is one of the following constants: OUTPUT_TYPE_DATASET, OUTPUT_TYPE_LPD, OUTPUT_TYPE_NONE, OUTPUT_TYPE_SYSOUT.

Notes:

- To execute this method, the isOutputTypeModifyAuthorized method must return True.
- This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.
- If this method is executed before the setOUDName method, then it is applied to the default output descriptor. Therefore to apply it to another output descriptor, execute the setOUDName method before executing this method.

isOutputTypeModifyAuthorized

Method: isOutputTypeModifyAuthorized()

Returns: boolean. True, if the logged-on user is authorized for this action. False, otherwise.

setRangeForSingleDoc

This method is only applicable when printing a single document. There is a limit of 132 characters. When connected to an RSD Folders Server running on a z/OS platform, there is a limit of 16 ranges.

Method: setRangeForSingleDoc(String range)

Parameter: "range" is the range of pages to extract. To indicate more than one range, separate each range with a comma.

For example to extract pages 4-7, 9 and 11-14, the range would be specified as: 4-7, 9, 11-14.

setUrgent

Method: setUrgent(boolean urgent)

Parameter: "urgent" if True, the extraction is marked as urgent. If False, this status is not applied.

Note: To execute this method, the isUrgentModifyAuthorized method must return True.

isUrgentModifyAuthorized

Method: isUrgentModifyAuthorized()

Returns: boolean. True, if the logged-on user is authorized for this action. False, otherwise.

Get methods

getOUDDescription

Method: getOUDDescription()

Returns: String. The description for the currently set output descriptor.

getOUDName

Method: getOUDName()

Returns: String. The name for the currently set output descriptor.

getOutputType

Method: getOutputType()

Returns: int. One of the following constants: OUTPUT_TYPE_DATASET, OUTPUT_TYPE_LPD, OUTPUT_TYPE_NONE, OUTPUT_TYPE_SYSOUT.

getRequestNumber

Method: getRequestNumber()

Returns: int. The number of the request on the server. This method is only applicable if connected to an RSD Folders Server running on a z/OS platform.

isAFPInterpreted

Method: isAFPInterpreted()

Returns: boolean. True, if AFPInterpreted is set to True. False, otherwise.

isCollate

Method: isCollate()

Returns: boolean. True, if collate is set to True. False, otherwise.

isInternalPrintCharacteristics

Method: isInternalPrintCharacteristics()

Returns: boolean. True, if InternalPrintCharacteristics is set to true. False, otherwise.

isUrgent

Method: isUrgent()

Returns: boolean. True, if urgent is set to true. False, otherwise.

Retrieve a list of output descriptors

The **FLDOUDList** object provides a list of output descriptors.

Constructor: FLDOUDList(FLDConnection connectObject) **

Constructor: FLDDocumentInfo(FLDDocumentList documentList) **

Constructor: FLDFolderList(FLDSetFilter setFilter) **

Parameters:

- "connectObject" is a valid instance of the FLDConnection object.
- "documentList" is a valid instance of the FLDDocumentList object. For more information, see "Retrieve a list of documents" on page 56.
- "setFilter" is a valid FLDSetFilter object. For more information, see "Set a filter" on page 38.

To restrict the list

These methods are only applicable when connected to an RSD Folders server running on a z/OS platform.

setExtractStatus

Method: setExtractStatus(int status)

Parameter: "status" is one of the following constants: STATUS_DYNAMIC_EXTRACT, STATUS_ EXTRACT, STATUS_LPD, STATUS_SYSOUT, STATUS_DATASET.

See notes following.

Notes:

- More than one status can be set. For example, the list can be restricted to output descriptors that specify a dynamic extract only for LPD output.
- To specify more than one status, use the 'pipe' character. The following, when executed implements the above example. setExtractStatus(STATUS_DYNAMIC_EXTRACT | STATUS_LPD).

getExtractStatus

Method: getExtractStatus()

Returns: int. The status set using the setExtractStatus method.

Access a record in the list

The following methods navigate forward in a list of output descriptors.

The methods return boolean. True, if a record can be retrieved. False, otherwise.

readFirst

Positions at the first record in the list.

Method: readFirst() **

readNext

Positions at the next record in the list. Assumes a readFirst has been executed.

Method: readNext() **

Get the information in the record

getOUDListEntry

This method implements an instance of the **FLDOUDListEntry** interface. which contains the methods getOUDDescription and getOUDName (documented below).

Method: getOUDListEntry()

getOUDDescription

Method: getOUDDescription()

Returns: String. The description for the output descriptor.

getOUDName

Method: getOUDName()

Returns: String. The name of the output descriptor.

Error messages

Error messages, generated while using the RSD Folders Java API, can be read using methods associated with the FLDException object.

Topics covered:

- "Objects providing information on exceptions" on page 125
- "List of error messages" on page 126

Objects providing information on exceptions

There are two objects that retrieve "exception" information. For more information on exceptions, see "Exceptions and error messages" on page 14.

FLDException

This object is automatically created as needed when an exception is generated.

Method: getNR()

Returns: int. The number associated with the error.

FLDExceptionMsg

This object is accessed using the method "getFLDExceptionMsg" discussed on page 24.

Method: getNR()

Returns: int. The number associated with the error.

Method: getMessage()

Returns: String. A generalized error message.

Method: getLocalizedMessage()

Returns: String. The detailed error message.

Generic example to retrieve an error message

```
Try {
connectInfo = new FLDConnection();
// create objects
// use associated methods
}
catch (FLDException e) // an error occurs
{
FLDExceptionMsg msg=connectInfo.getFLDExceptionMsg(e);
// print the error messages and number
System.out.println("Error message : "+msg.getNr()+" "+msg.getMessage()+"
"+msg.getLocalizedMessage());
}
// close objects
// close connection to the server
```

List of error messages

Following are the error messages are grouped by topic. For each topic, information is provided as to the object generating the error.

The information on the first general group of errors is returned when the getMessage() method is applied. The second general group of errors is returned when the getLocalizedMessage() method is applied.

Messages returned by the API

These messages are returned using the getMessage method. When they errors are returned, additional information may be returned by using the getLocalizedMessage() method. This is particularly true for error # 0000000.

Error Number	Information
0000000	Error returned by server. For more information, see "Messages returned by the server" on page 127.
1001000	Connection error.
1001001	Server and client Versions don't match.
1001002	Connection already active.
1002000	Invalid parameter.
1002001	Input parameter too long.
1002002	Use of null or invalid reference object.
1002003	Use of invalid or closed object.
1002004	Use of invalid negative value.
1002005	Invalid input parameter.
1002007	Incompatible with this version.
1003000	Encoding error.
1004000	Error creating ascii/ebcdic translation table.
1005000	Communication error.
1005001	Invalid packet size.
1005002	Invalid CH.
1005003	Invalid checksum.
1005004	Invalid date format.
1006000	Error reading text and message table.
1007000	License ended or invalid.
1008000	Operation not allowed.
1008001	Security error.
1008002	No transfer for microfiche document.
1008003	Modification of setting, to filter on matching or not matching folder names, not allowed.
1008004	Modification of setting, to filter on matching or not matching document names, not allowed
1008005	Folder name is not consistent with the filter definition.
1008007	Folder type is not consistent with the filter definition.
1008006	Document name is not consistent with the filter definition.
1008008	Document type is not consistent with the filter definition.
1009000	System error.

Messages returned by the server

These messages are returned using the getLocalizedMessage method. For all of the following, the getMessage method returns “Error returned by server”.

General error messages

These messages can be generated for all API predefined objects and methods.

Error Number	Information
xxx064	Action denied by Folders server user routine.
xxx083	Environment not responding.
xxxx97	Database full.
xxx100	Maximum number of subtasks reached.
xxx101	Document deleted, action denied.
xxx147	Function not supported.
xxx151	Database is full.
xxx152	Access error on database.
xxx153	Unknown error received from the remote server.
xxx255	Error in question submitted to the server.

Errors associated with logon, identification and setting the environment

These errors are associated with the FLDConnection object. For more information, see "Initiate server/client connection" on page 18. Also see errors 241xxx which are associated with long identification.

Error Number	Information
000001	Maximum number of connections exceeded.
000002	There is not enough memory or resources on the server.
000003	Login again using alternative application.
000004	Communication protocol not accepted.
000005	Product will expire in xx days.
000008	Product expired.
000009	Product not installed.
000010	CPU not authorized.
000011	Rejected by user exit.
000015	Profile file MVS error.
001001	Access security system password mandatory.
001002	User name not defined.
001003	Invalid password.
001004	Expired password.
001005	Invalid new password.
001006	User revoked.
001007	Terminal not authorized for this user.
001008	Application not authorized for this user.
001009	Invalid Security Group Name.
001010	No function authorized.
001011	Server Security system: Access denied.
001012	Logon not available.

Error Number	Information
001013	User already connected.
001014	Maximum connection(s) reached.
001015	Access error to RSDCTL.
001016	Error on Profile.
001017	User rejected by user exit.
001018	Terminal rejected by user exit.
001019	Terminal rejected by user exit with no retry.
001053	Server access error. No retry allowed.
001080	Logon rejected.
002001	Environment not defined.
002002	Profile error
002003	Environment not available.

Error associated with verifying a folder type and document type

The first error is associated with the FLDFolderTypeList object (page 34) and the second with the FLDDocumentTypeList object (page 35).

Error Number	Information
006002	Folder type not found.
008002	Document type not found.

Error associated with setting the filter

This error is associated with the FLDSetFilter object. For more information, see "Set a filter" on page 38.

Error Number	Information
017002	Unknown filter name.
020002	Filter value not accepted by the server.

Errors associated with transferring resources

These errors are associated with either the FLDResourceTransfer object. For more information, see "Transfer document resources" on page 118.

Error Number	Information
025033	The resource database is not available.
025034	Open error on resource database.
025035	Access error to the database.
025036	AFP resource not found.
025037	Unknown AFP resource type.
025038	Record too large for buffer.

Errors associated with transferring a text or AFP document

These errors are associated with the FLDTextDocumentTransfer or FLDAFPDocumentTransfer object. For more information, see "Transfer a text document" on page 109 and see "Transfer an AFP document" on page 115.

Error Number	Information
026002	End of document. No record found.
026003	The document is not located in direct access media.

Error Number	Information
026004	Archive no longer exists.
026005	Read error on archive.
026006	The document is not in direct access. Demand confirmation.
026007	The document is not in direct access. Confirmation not demanded.
026009	Unavailable system resource.
026016	Environment error.
026017	Request denied by operator.
026018	Allocation failed. Unexpected error.
026033	Unavailable resource database.
026034	Resource database open error.
026035	I/O error on resource database.
026036	AFP resource not found.
026148	Error on work file.
026149	Find backwards error.
026150	Bottom of document processing error.
026151	Change archive error.

Errors associated with finding information in a document

These errors are associated with the FLDTextDocumentFind object. For more information, see "Restrict information in a text document" on page 112.

Error Number	Information
027001	Not found. End of range.
027002	Not found. End of document.

Errors associated with transferring a binary document

These errors are associated with the FLDBinaryDocumentTransfer object. For more information, see "Transfer a binary document" on page 114.

Error Number	Information
028003	The document is not located on direct access media.
028004	Archive no longer exists.
028005	Read error on archive.
028006	The document is not located on direct access. Demand confirmation.
028007	The document is not located on direct access. Confirmation not demanded.
028009	Unavailable system resource.
028016	Environment error.
028017	Request denied by operator.
028018	Allocation failed. Unexpected error.
028148	Error on work file.
028149	Find backwards error.
028150	Bottom of document processing error.
028151	Change archive error.

Errors associated document information

This error is associated with either the FLDDocumentInfo object. For more information, see "Technical document information" on page 65.

Error Number	Information
029002	Document not found.

Errors associated with print characteristics

These errors are associated with either the FLDDocumentResourceList object. For more information, see "Retrieve a document resource list" on page 116.

Error Number	Information
031001	No output descriptor for this document.
031002	No FCB in the output descriptor.
031004	FCB open library error
031005	FCB module not found in the library.
031006	FCB load module error.
031033	Resource database not available.
031034	Resource database open error.
031035	Resource database I/O error.
031036	AFP resource list not found.
031037	Xerox/DSC document: transfer must be performed one at a time.

Errors associated with an descriptor list

These errors are associated with the FLDDescriptorList object. For more information, see "Retrieve a descriptor list" on page 52.

Error Number	Information
032002	End of list. No document descriptor found.

Errors associated with uploading a file

These errors are associated with the FLDUploadFile object. For more information, see "Upload a file" on page 53.

Error Number	Information
033001	The document doesn't match the filter.
033002	Document not found.
033003	Descriptor not found.
033151	RPI database full.
034002	Checksum error.
034097	The RPI database is full.
035002	Checksum error.
035003	Descriptor not found.

Errors associated with renaming a document

These errors are associated with the FLDDocumentListEntry object. For more information, see "FLDDocumentListEntry interface" on page 60.

Error Number	Information
039001	The document name is inconsistent with the filter, rename not accepted.
039002	Document not found.

Errors associated with renaming a folder

These errors are associated with the FLDFolderListEntry object. For more information, see "Get the information in the record" on page 49.

Error Number	Information
040001	The folder name is inconsistent with the filter, rename not accepted.
040002	Folder not found.
040003	The folder name already exists, rename not accepted.

Errors associated with creating a folder

These errors are associated with the FLDCreateFolder object. For more information, see "Create a folder" on page 37.

Error Number	Information
041003	Cannot create the folder record.
041004	Invalid folder name: contains * or ?.
041097	RPI database full.

Errors associated with version management

These errors are associated with the FLDVersionList object. For more information, see "Manage document versions" on page 76.

Error Number	Information
046071	No version found
046072	Version already locked
046074	Last existing version cannot be deleted
046075	Document already versioned
046076	No lock found
046077	Currently locked by another user
046078	Version locked, action denied
046079	Version is not the active one, cannot be locked

Errors associated with version labels

These errors are associated with the FLDVersion interface. For more information, see "Manage a specific version of a document" on page 78.

Error Number	Information
046120	Label value already exists
046121	Label value not found

Errors associated with extracting a folder or a document

These errors are associated with the FLDExtract interface. For more information, see "Extract a folder or document" on page 120.

Error Number	Information
048001	OUD not found.
048002	Page number is greater than the document size.
048003	Page number syntax error.
048004	No folder found.
048005	Invalid document key.
048006	No archive for this run.
048007	No folder type given.
048008	Invalid OUD.
048009	Extract queue is full.
048016	I/O Error on extract queue.
048017	Open error on extract queue.
048018	Open error on I/O module.
048019	Invalid page range value
048020	Page range too large.
048021	Not in sequence.
048022	Page range is greater than maximum page number.
048023	Invalid field format.
048023	RPD open error.
048025	RPD I/O Error.
048032	No PDB found.
048033	End of document.
048034	Display document not authorized.
048035	Archive/Workfile open error.
048036	Archive access I/O error.
048037	Document on tape direct loading.
048038	Unavailable system resource.
048039	Environmental Error.
048040	Request denied.
048041	Alloc failed, unexpected error.
048048	AFP resources DB not available.
048049	AFP resources DB open error.
048050	AFP resources DB I/O error.
048051	AFP resources (FormDef) not found.
048052	Dynamic extract failed, unexpected RC.
048053	Doc on tape, Workfile I/O error.
048054	Bottom of document procedure error.
048055	Change archive error.
048056	Access to RPI DB error.
048057	Other error.
048065	Dynamic extract failed, error on ASTRPLIB.
048066	Dynamic extract failed, print format not found.

Error Number	Information
048067	Dynamic extract failed, output alloc failed.
048068	Dynamic extract failed, AFP resource cannot be found.
048069	Invalid Class Name.
048070	Invalid Copy Number.
048071	Invalid Destination.
048072	Invalid Room.
048073	Invalid Form.
048080	Invalid FCB.
048081	Invalid Writer Name.
048082	Invalid Destination 2.
048083	Invalid maximum number of chars per line.
048084	Invalid maximum number of lines per page.
048085	Invalid LPD server address or port.

Errors associated with accessing a document using the doc id

These errors are associated with the executing the method getFLDDocumentByID. For more information, see "Direct access to document information" on page 67.

Error Number	Information
061001	Document does not exist anymore.
061002	Document not found.
061003	Document does not match filter.
061004	Document type not authorized.
061005	Folder type not authorized.

Errors associated with changing the status of a mark

These errors are associated with the FLDDocument interface. For more information, see "General document information" on page 60.

Error Number	Information
090001	User not authorized to change the mark status.
090002	Document not found.
090003	Document already marked with that status.
090004	Mark type not valid for that document type.

Error associated with retrieving a list of mark types

This error is associated with the FLDMarkList object. For more information, see "Retrieve a list of mark types" on page 36.

Error Number	Information
091002	End of list. No mark type found.

Error associated with reading an index field list

These errors are associated with the FLDIndexFieldList object. For more information, see "Retrieve a list of index field descriptors" on page 92.

Error Number	Information
096001	Function not allowed for this user.
096008	ABI component license has expired.
096009	DB server not started.
096010	Online server and DB server referencing different databases.

Error associated with displaying a document list based on a query

These errors are associated with the FLDQueryList object. For more information, see "Retrieve a list of documents based on an index query" on page 95.

Error Number	Information
097001	Function not allowed for this user.
097002	No documents found.
097003	Query not valid.
097004	Unknown field name.
097005	Error in query.
097006	DB server error.
097007	Request did not succeed. Timeout while processing.
097008	ABI component license has expired.
097009	DB server not started.
097010	Online server and DB server referencing different databases.

Error associated with updating an index entry in the ABI

These errors are associated with the FLDIndexFieldUpdate interface. For more information, see "Add, delete or change index entries" on page 103.

Error Number	Information
098001	User not authorized for this action.
098002	Unknown value.
098003	Error during update.
098004	Index field unknown.
098005	Invalid value.
098006	Value already exists.
098008	License to use ABI has expired.
098009	DB server not running.
098010	Online server and DB server referencing different databases.

Errors associated with logon, identification and setting the environment

These errors are associated with the FLDConnection object when connecting to server version 4.1. For more information, see "Initiate server/client connection" on page 18.

Error Number	Information
241001	Access security system password mandatory.
241002	User name not defined.
241003	Invalid password.
241004	Expired password.
241005	Invalid new password.
241006	User revoked.
241007	Terminal not authorized for this user.
241008	Application not authorized for this user.
241009	Invalid Security Group Name.
241010	No function authorized.
241011	Server Security system: Access denied.
241012	Logon not available.
241013	User already connected.
241014	Maximum connection(s) reached.
241015	Access error to RSDCTL.
241016	Error on Profile.
241017	User rejected by user exit.
241018	Terminal rejected by user exit.
241019	Terminal rejected by user exit with no retry.
241020	LDAP error.
241021	Support for document ID not activated.
241022	User already connected.
241023	Forced identification failed.
241080	Logon rejected.

Errors associated with document history

These errors are associated with the FLDDocumentHistoryList object. For more information, see "Manage document history" on page 82.

Error Number	Information
1046061	User name is too long.
1046050	Document history is not supported by the server.
1046056	Invalid document history action returned by the server.

Index