Programmer's Guide

Release 3.2.1

April 2006

FileNet Corporation
3565 Harbor Boulevard
Costa Mesa, California 92626
1.800.FILENET (345.3638)
Outside the U.S., call:
1 .7 1 4 .3 2 7.3 4 0 0
www.filenet.com

Notices

Contents

About This Guide

Integration with existing Enterprise Information Systems (EIS) is the key to success in businesses moving towards an e-business strategy.

The Java 2 Enterprise Edition (J2EE) Connector Architecture defines a standard architecture to integrate the J2EE platform with heterogeneous EISs.

Image Services Resource Adapter (ISRA) is a system-level software driver that can be used by a Java application component to connect to the FileNet Image Services (IS). It is compliant to the J2EE Connector architecture v1.0.

ISRA provides an alternative to IDM Web Services for IS customers. In addition, it provides a Web solution that does not require Microsoft technology or product support.

Overview

This guide is designed to assist programmers in using ISRA. The following is an overview of information contained in this guide:

Chapter 1 – **ISRA Overview:** Introduces and explains the ISRA architecture.

Chapter 2 – **Using the Common Client Interface (CCI):** Explains how to obtain an ISRA connection, create the required objects, and invoke interactions.

Chapter 3 – **Supported ISRA Interactions:** Describes the various interactions supported by ISRA.

Appendix A: Contains references of miscellaneous classes used in conjunction with ISRA.

Appendix B: Contains XML schema for Image Manager Annotations.

Index: Contains listing of key terms used in this guide for easy reference; sorted in alphabetical order to help locate appropriate information easily.

Target Audience

The ISRA Programmer's guide is designed for software developers with working knowledge of:

- Java Technology.
- J2EE Connector Architecture specification released by Sun Microsystems.
- FileNet Image Services (IS) 3.6 SP2 or above.

Conventions Used In this Guide

This guide uses the following type, text, and naming conventions:

Convention	Description
Italics type	Indicates referenced section name or document title.
`Fixed Size`	Indicates code samples, syntax, class names, and parameters.
Comments in Code	Indicates code sample comments. Typically, comments appear within code sample blocks.
Blue text	Indicates a link to another topic, a link to another section in the same topic, or a link to an external topic.
Note	Includes information that one might find useful or would want to know about in the given context.
Tip	Includes information one should treat as a general guideline while developing.

Education

For details on FileNet's Global Learning Services, please visit the Service & Support area at www.filenet.com.

Related References

For all ImageViewer parameters, please refer to the FNImageViewer documentation after installing ISRA 3.2.1. The path for FNImageViewer documentation is:

<ISRA-home>\ISRA321\FNImageViewer\docs

For all P8 System Manager related information, please refer to P8 System Manager documentation after installing ISRA 3.2.1. The path for P8 System Manager documentation is:

<ISRA-home>\ISRA321\SystemManager\docs

Comments and Suggestions

FileNet invites all customers to communicate with the Documentation group on any question or comment related to FileNet manuals and online help. Your suggestions will help us deliver better.

1. ISRA Overview

ISRA is a system-level software driver, compliant with the J2EE Connector Architecture v1.0, to connect to FileNet Image Services (IS) using a Java application component or client.

The key features of ISRA are:

- It supports the system contracts, specified in the Connector Architecture specifications: Connection Management and Security.

Note: ISRA does not support Transaction management contract.

- It provides Common Client Interface (CCI) that can be used as the primary interface to interact with the IS.
- It can be used in, both, managed and non-managed environments.

Note: ISRA is currently supported for production deployment, only in the Managed Environment. You can use ISRA, in Non-managed Environment, for development.

- Communicates directly with IS and does not depend on any client side IS connectivity like the Image Services Toolkit.

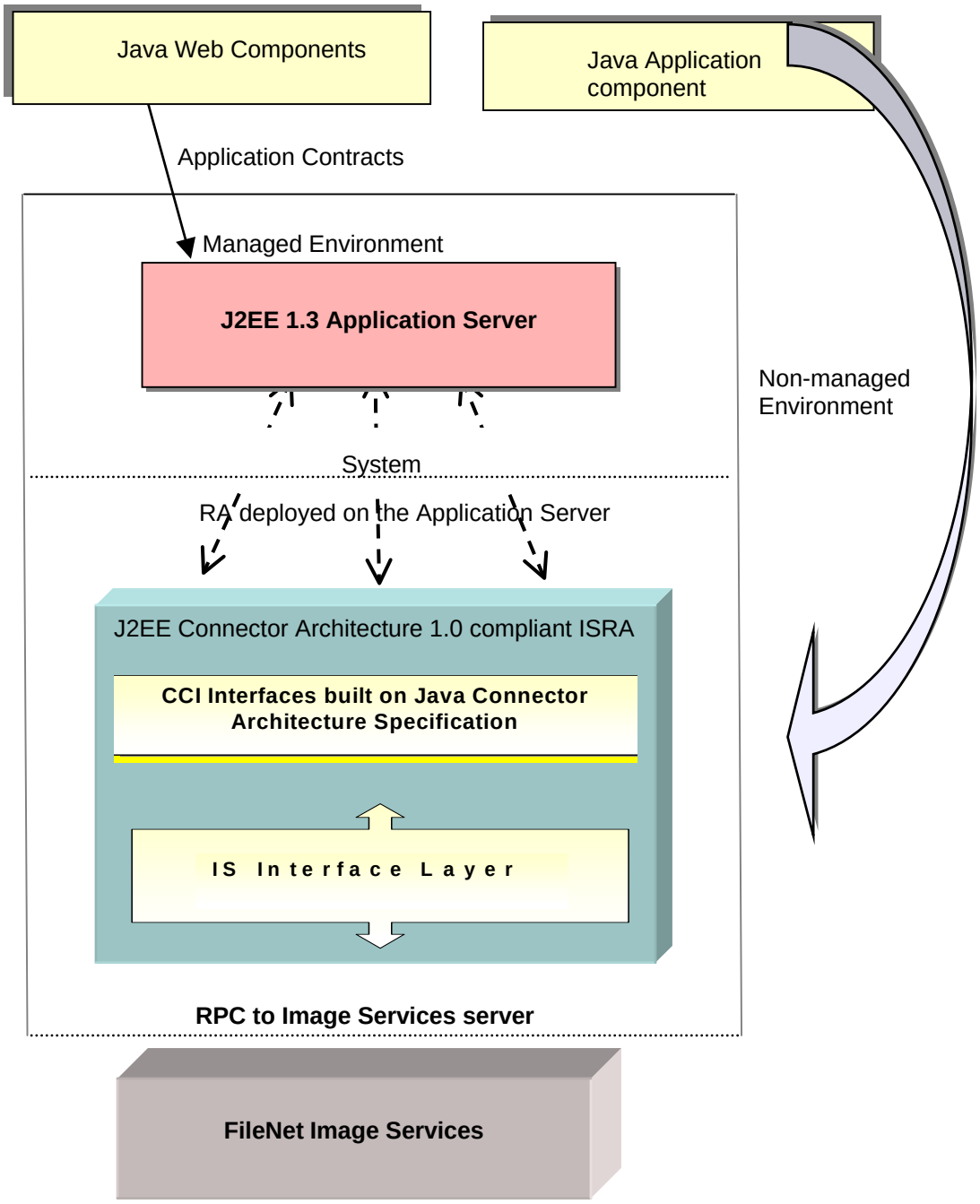
ISRA provides these functionalities:

- Query: Searching for index data of one or more documents.
- Document Retrieval: Retrieving document content (pages).
- Add documents: Committing a new document into the IS repository.
- Document properties: Retrieving and updating document properties, and canceling document properties update.
- Meta-data: Retrieving document class indices, menu description and security info.
- Delete documents: Deleting document(s) from the IS repository.
- Folder Browsing: Retrieving folders, folder attributes and documents within a folder.
- Folder Creation, Deletion and Update: Creating folders, deleting existing folder and updating the properties of the existing folders.
- Document filing and removal: Filing and un-filing document(s) into and from a folder.
- Get document folders: Finding folders (the location) where the document is filed.
- Queue support: Retrieving queue names and user field description, creating workspaces and queues, and, inserting, deleting and updating queue entries.

- Annotation support: Retrieving, creating, deleting and updating annotation details.
- Password Related: Retrieving password status and changing password of a user.
- Printing/Faxing: Printing and faxing documents existing in IS.
- Cache List: Retrieving the list of caches configured in IS.

ISRA Architecture

The Following figure illustrates the ISRA architecture:



ISRA Usage Environments

ISRA can be used by application components in the following two ways:

Managed Environment

In a managed environment, ISRA is deployed on an application server. Both, the application server and ISRA, collaborate to provide basic services like connection pooling and security. An application component does a Java Naming and Directory Interface (JNDI) lookup on the application server to get a `javax.resource.cci.ConnectionFactory` instance. This is a multi-tier scenario, where the application component interacts with the application server, and the application server interacts with ISRA. ISRA is managed within the address space of the application server.

Non-Managed Environment

In a non-managed environment, an application component can, either, use the ISRA library directly, or use JNDI registration mechanism to locate the `ConnectionFactory`. In the former case, ISRA acts as a client Application Program Interface (API) and the application component creates a `javax.resource.cci.ConnectionFactory` instance on its own. In the latter case, application component does a lookup into the namespace of a JNDI service to get a pre-configured `ConnectionFactory` instance.

Note: ISRA does not support any connection pooling mechanism in a non-managed Environment.

System Contracts

The system contracts link ISRA to important services that are managed by the application server. They keep all system-level mechanisms transparent from the application components. This allows an application component provider to focus on the development of business and presentation logic, and defends from system-level issues related to Enterprise Information Server (EIS) integration. This promotes fast and easy development of scalable enterprise applications.

The system contracts specified in the Connector Architecture specifications are:

- Connection Management Contract.
- Transaction Management Contract.
- Security Contract.

Connection Management Contract

Allows an application server to pool connections to the EIS, and allows application components to connect to the EIS. Connection pooling is transparent to the application components.

In ISRA, the Connector Architecture specification of 'physical connection' maps to the IS logon session. The application component uses the `javax.resource.cci.Connection` as the application level connection handle to execute interactions.

Connection Pooling and Connection Sharing in ISRA

- Connection Pooling.

An application component uses the `ConnectionFactory` interface to get an application `Connection` instance. ISRA acts as a factory of connections; each connection resolves allocation of some physical or network resources. Creating and destroying connections to the IS is, both, time-consuming and cost-ridden. Thus, ISRA allows the application server to manage and pool connections through connection pooling. Connection pooling leads to better scalability and performance in a managed environment. The implementation of connection pooling algorithm is application server specific.

ISRA supports requisite Service Provider Interface (SPI) interfaces and provides SPI implementation classes. SPI is a Connection Management Contract that provides support for connection pooling.

- Connection Sharing.

  Managed Environment

  With ISRA, multiple clients can share a physical connection, if they have the same credentials.

  Whenever a new user (if there are no users logged on with the same credentials) requests a connection to IS, a new connection is created.

  If another user requests a connection to IS, either of these actions takes place:

  - `If the User has the same credentials as an existing one, a physical connection associated with the existing user is used, instead of creating a new physical connection.`
  - `If the User has different credentials, a new physical connection is created.`

  Non-Managed Environment

  In a non-managed environment, ISRA does not maintain a connection pool, but the connection sharing mechanism is similar to that of the managed environment.

Transaction Management Contract

This contract is between an application server and an EIS that supports transactional access. ISRA does not support transactions. An invocation of either `getXAResource()` or `getLocalTransaction()` method on the `Connection` Object throws a `javax.resource.NotSupportedException` exception.

Security Contract

This contract enables a secure access to the IS server. It provides support for a secure application environment.

The security contract between the application server and ISRA extends the connection management contract by adding security specific details.

The security contract supports sign-on to IS by:

- Propagating the security context from the application server to ISRA.
- Passing the connection request from ISRA to the application server.

Note: ISRA does not support re-authentication of connection request.

Depending on whether the application server or the application component is managing IS sign-on; the connection can be created in these two ways:

- **Container Managed Sign-on:** The application components do not pass any security information in the `getConnection()` method and the application server takes the responsibility of managing the sign-on to the IS**.**
- **Component Managed Sign-on**: The application components provide explicit information in the `getConnection()` method, by passing the security information through the `javax.resource.cci.ConnectionSpec` object.

Common Client Interface (CCI)

ISRA supports the Connector Architecture Common Client Interface (CCI) API. CCI is a set of Java interfaces that allow an application component to perform IS operations.

CCI facilitates access to IS from application components such as Enterprise Java Beans (EJB), Java Server Pages (JSP), Servlets and etc. The steps involved in accessing IS through ISRA are:

1. Application component establishes a connection to the IS using the `ConnectionFactory.`

Note: `Connection` object represents the application level connection handle to the IS, and is used for subsequent interactions with the IS.

2. Application component performs interactions with the IS, such as retrieving the document's content, using an `Interaction` object.

Note: The application component defines the specifications of the `Interaction` object using an `InteractionSpec` object.

3. The application component retrieves data from the IS, such as Document Index Records, using a `javax.resource.cci.Record` instance. This Record instance can be a `javax.resource.cci.MappedRecord`, `javax.resource.cci.IndexedRecord`, or a `javax.resource.cci.ResultSet`.

CCI Interfaces

CCI interfaces can be categorized as:

- Connection Interfaces
- Interaction Interfaces
- Record Interfaces

Connection Interfaces

The interfaces for the connection factory and application level connection are:

- **ConnectionFactory:** The `javax.resource.cci.ConnectionFactory` acts as a factory of connections and provides the application component with a `Connection` instance to IS.

The `ConnectionFactory` interface methods are:

  - `getConnection() returns a connection to IS. The arguments include user name and password.`
  - `get RecordFactory() returns a javax.resource.cci.RecordFactory instance.`

- **Connection:** The `javax.resource.cci.Connection` is a connection handle to IS**.**

  The `Connection` Interface methods are:

  - `createInteraction(): creates and returns a javax.resource.cci.Interaction object to perform interactions on IS.`
  - `close(): closes the connection to the IS.`

- **ConnectionSpec:** The `javax.resource.cci.ConnectionSpec` instance is used by an application component to pass connection request-specific properties to the `getConnection()` method of `ConnectionFactory`. It

carries the user name and password (and also an additional optional parameter, loginType, used for LDAP authentication) from the application component to ISRA.

Interaction Interfaces

The interfaces that enable a component to drive an interaction with the IS are:

- **Interaction:** The `javax.resource.cci.Interaction` allows the application component to execute IS functions**.**

The Interaction interface provides two variants of `execute` method, to the application component, to invoke supported interactions. These variants are:

  - `execute()` method that takes an input Record, output Record and a `javax.resource.cci.InteractionSpec`: executes the IS function represented by the `InteractionSpec` and updates the output Record.

  - `execute()` method that takes an input Record and an `InteractionSpec`: executes the IS function represented by the `InteractionSpec` and returns the output Record.

- **InteractionSpec:** The `javax.resource.cci.InteractionSpec` defines an interaction with the IS, including function name and function-specific parameters.

Note: ISRA provides implementation classes for the `ConnectionSpec` and `InteractionSpec` interfaces in `ISRA.jar` as well as `client_helper.jar`. Application components should use these classes to get a connection to the IS and to define interactions, respectively. In a Managed Environment, `client_helper.jar` needs to be included in the application server's classpath. See the Appendix A: References for more details on `ConnectionSpec` and `InteractionSpec` interfaces.

Record Interfaces

A `Record` is the Java representation of a data structure used as input or output to an IS function.

- **RecordFactory:** The `javax.resource.cci.RecordFactory` is used to create `MappedRecord` and `IndexedRecord` instances. It provides the application with a `Record` instance for data transfer between application component and ISRA.

  Methods of the `RecordFactory` interface are:

  - `createIndexedRecord()`: creates an ordered collection of objects.

  - `createMappedRecord()`: creates an object of key-value pairs.

- **Record:** The `javax.resource.cci.Record` instance is used as an input or output to the execute methods defined in Interaction. It is the base interface for the different kinds of record instances:
  - `MappedRecord`: `stores data in the form of key-value pairs.`
  - `IndexedRecord`: `represents an ordered collection of elements based on` `java.util.List` `interface.`
  - `ResultSet`: `represents data in a tabular form based on` `java.sql.ResultSet`.

2. Using the Common Client Interface (CCI)

This chapter explains how to:

- Use CCI interfaces and ISRA CCI implementation classes
- Obtain a Connection.
- Create an Interaction Object.
- Create a Record Object.
- Invoke an Interaction.

Obtaining a Connection

You can obtain a connection to ISRA in a:

- Managed Environment, or
- Non-Managed Environment.

Managed Environment

To obtain a connection in a Managed Environment, look up for the `ConnectionFactory` interface in the JNDI name space and establish a connection using it.

Looking Up the ConnectionFactory

JNDI is an interface used by an application component to access naming and directory services. The application server takes care of binding ISRA `ConnectionFactory` object to its JNDI namespace.

Application component performs a JNDI lookup to acquire a reference to the `ConnectionFactory` instance.

The code snippet shows how to look up the `ConnectionFactory`. It assumes that the `ConnectionFactory` object is bound to the JNDI of the application server, with the name “ISCF”.

```
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Connection;
```

```
import javax.resource.cci.MappedRecord;
import javax.resource.cci.IndexedRecord;
import javax.resource.cci.ResultSet;
import javax.resource.ResourceException;
//Get the InitialContextFactory.
Context context = new InitialContext();
//Perform Lookup
ConnectionFactory connectionFactory = (ConnectionFactory)
context.lookup ("ISCF”);
```

Establishing A Connection Using The ConnectionFactory

After acquiring a reference to ISRA `ConnectionFactory` instance, the application component calls the `getConnection` method of the `ConnectionFactory`. There are two variants of this method, one using a `ConnectionSpec` and the other without using it.

Using ConnectionSpec

This is a Component Managed Sign-on scenario, where the application component calls the `getConnection` method, and passes the `ConnectionSpec` instance as an input parameter.

The code sample shows how to establish a connection using the `ConnectionSpec`:

```
FN_IS_CciConnectionSpec connectionSpec = new
com.filenet.is.ra.cci.FN_IS_CciConnectionSpec();
connectionSpec.setUserName(“operator”);
connectionSpec.setPassword(“op_password”);
Connection connection = null;
try{
      connection =
connectionFactory.getConnection(connectionSpec);
}catch(ResourceException re){
      System.out.println(“Failed to establish
      connection:” + re.getMessage();
}
```

Another code sample shows how to establish a connection using the three-parameter method of `ConnectionSpec`:

```
FN_IS_CciConnectionSpec connectionSpec = new
com.filenet.is.ra.cci.FN_IS_CciConnectionSpec();
```

```
connectionSpec.setUserName("operator");
connectionSpec.setPassword("op_password");
/* 0 – Direct IS Login and 1 – Authentication through
LDAP Server, specified in Deployment Descriptor, before
IS login.*/
connectionSpec.setLoginType(newInteger(0));
Connection connection = null;
try{
      connection =
connectionFactory.getConnection(connectionSpec);
}catch(ResourceException re){
      System.out.println("Failed to establish
      connection:" + re.getMessage();
}
```

Note: When a user tries to get a Connection object with invalid credentials for *max-capacity* number of times, WebLogic Server (WLS) throws an exception. But some versions of the WLS do not throw any exception for the next (max-capacity+1) request, and returns `Connection` object as null. However, they print an error message in the server console. The code to avoid `java.lang.NullPointerException`, in the above scenario is:

```
//Check for null and proceed.
If (connection != null){...}
```

Without using ConnectionSpec object

This is a Container Managed Sign-on scenario, where the application component does not create any `ConnectionSpec` object. Instead, it calls the `getConnection` method, with no arguments. The application server uses the default set of user credentials that are already configured in the application server. The required users and roles are created in application server, and the mapping of application server's roles and IS users/password is created in application server's deployment descriptor.

The code sample shows how to establish a connection without using the `ConnectionSpec` object:

```
Connection connection = null;
try{
connection = connectionFactory.getConnection();
}catch(ResourceException re){
System.out.println("Failed to establish connection:" +
re.getMessage();
}
//Check for null and proceed.
```

```
if(connection != null){
...
...
}
```

Non-Managed Environment

ISRA supports access to IS server from non-managed application components, like standalone Java applications. In a non-managed two-tier application environment, an application component directly uses the ISRA library. This model is similar to a two-tier Java Database Connectivity (JDBC) application component accessing a database system.

In a non-managed scenario, the application developer can also use a programming model, similar to the managed scenario. The non-managed (using JNDI) case involves looking up a `ConnectionFactory` instance in JNDI namespace, getting an IS connection, using the connection to access the IS, and finally closing the connection.

In a non-managed scenario, the `ConnectionManager` implementation is provided by ISRA (as a default `ConnectionManager` implementation).

Setting Up a Non-Managed Environment

The code sample that uses ISRA to interact with the IS, in a non-managed environment, without JNDI is:

See Appendix A: References for the detailed information of the methods of FN_IS_SpiManagedConnectionFactory class.

```
import com.filenet.is.ra.cci.*;
import com.filenet.is.ra.spi.*;
import javax.resource.cci.*;
import javax.resource.spi.*;
import javax.resource.ResourceException;
FN_IS_SpiManagedConnectionFactory
managedconnectionFactory;
ConnectionFactory connectionFactory;
Connection connection;
Interaction interaction;
FN_IS_CciConnectionSpec connectionSpec;
FN_IS_CciInteractionSpec interactionSpec;
RecordFactory recordFactory;

try {
```

```
// Create a ManagedConnectionFactory instance:

managedconnectionFactory = new
FN_IS_SpiManagedConnectionFactory();

// Set the properties of the ManagedConnectionFactory.

managedconnectionFactory.setDomainName("IS_Domain");

managedconnectionFactory.setOrganizationName("FileNet");

managedconnectionFactory.setProductName("FileNetISRA");

managedconnectionFactory.setLogFileName(“FileName”);

managedconnectionFactory.setLoggingLevel(new

Integer(2));

managedconnectionFactory.setLoggingMode(new Integer(2));

managedconnectionFactory.setPageBufferSize(new
Integer(64));

/* Add these methods if LDAP authentication is required.

managedconnectionFactory.setLdapImplClassName
(“com.filenet.is.ra.fnis.FN_IS_IPlanetImpl”);

managedconnectionFactory.setLdapImplClassString
(“filenetserver;389;ou=filenet,ou=users,dc=filenetIS”);

managedconnectionFactory.setInherentLogin(new
Integer(1));

*/

/* Create a connection factory using the default
connection manager. */

connectionFactory =
managedconnectionFactory.createConnectionFactory();

// Set up ConnectionSpec.

connectionSpec = new FN_IS_CciConnectionSpec();

connectionSpec.setUserName(“operator”);

connectionSpec.setPassword(“op_password”);

/* Add this parameter if LDAP authentication is required.

connectionSpec.setLoginType(new Integer(1));

*/

// Get connection to IS.

connection =
connectionFactory.getConnection(connectionSpec);

// Set up Interaction.
```

```
interaction = connection.createInteraction();

FN_IS_CciInteractionSpec interactionSpec = new
FN_IS_CciInteractionSpec();
interactionSpec.setFunctionName("GetDocClassIndices");

interactionSpec.setInteractionVerb(InteractionSpec.SYNC_S
END_RECEIVE);

recordFactory = connectionFactory.getRecordFactory();


Record iRecord =
recordFactory.createMappedRecord(“DocClassName”);

Record oRecord =
recordFactory.createMappedRecord(“DocClassIndex”);

// Execute the interaction with the IS.

boolean success = interaction.execute(interactionSpec,
iRecord, oRecord);

if (success) {

processOutputRecord(oRecord);

}

// Close the interaction and the connection.

interaction.close();

connection.close();

}

catch (ResourceException re) {

...

}
```

Setting up a Non-Managed Environment Using JNDI

The application component can use the JNDI mechanism to lookup a `ConnectionFactory` object that is bound to the JNDI name space of a directory service. This mode of deployment is useful when many application components share a common `ConnectionFactory` object. This allows the application components to be transparent about the configuration details of the `ConnectionFactory`.

Note: The application components should have all the resource adapter classes in the CLASSPATH.

The code snippet for the application component performing a lookup for a connection factory is:

```
//Get the Context of the Directory service.
```

```
javax.naming.Context context =...;

/* Lookup the ConnectionFactory bound with name
“FileNet_ISRA_CF”. */

javax.resource.cci.ConnectionFactory connectionFactory =
(ConnectionFactory)context.lookup("FileNet_ISRA_CF");
```

To enable the JNDI lookup of a `ConnectionFactory`, some basic JNDI infrastructure normally supplied by the application servers, must be developed. Writing a utility class, which would be used to bind the ConnectionFactory to a JNDI namespace, can do this.

The two phases to be considered while looking the JNDI interface: object binding and object lookup. The binding phase takes place in a managed environment when a `ConnectionFactory` instance is created. In a non-managed environment, a utility program needs to run prior to the first lookup. In this environment, a `ConnectionFactory` instance is stored in a `javax.naming.Reference` instance, which is bound to the JNDI context.

The lookup phase causes the JNDI framework to call the `getObjectInstance` method on an `ObjectFactory` implementation, specified in the binding phase. To support the Reference mechanism in a non-managed environment, a helper class of ISRA `(FN_IS_ObjectFactory)` provides an implementation of `javax.naming.spi.ObjectFactory`, which defines the `getObjectInstance` method. The `getObjectInstance` method retrieves data from a Reference, which is passed into it (the same Reference that was bound earlier). Data stored in the Reference can be any number of objects that implement `java.io.Serializable`. The `getObjectInstance` method only needs to retrieve the `ConnectionFactory` and return it for JNDI to complete the object lookup.

Note: See Appendix A: References for details on how the Reference to a `ConnectionFactory` object can be bound to a JNDI service.

Creating the Interaction Object

To perform an interaction with the IS, the application component creates an `Interaction` object by calling the `createInteraction` method of the Connection object.

The code sample to create an Interaction Object is:

```
interaction = connection.createInteraction();
```

Creating the InteractionSpec Object

To define the specifications of the `Interaction` object, the application component creates a `javax.resource.cci.InteractionSpec` object. `InteractionSpec` implementation class provides getter and setter methods for all its supported properties.

The standard properties common to all interactions are:

- **FunctionName**: is a string representing the name of an EIS function. For example: FindDocuments is the function name used in FindDocuments interaction, to query documents in the IS.

  The valid function names are explained in the following table:

Types Of Interactions	Function Name	Description
Document Interactions	FindDocuments	Function Name used in FindDocuments interaction.
	GetDocumentContent	Function Name used in GetDocumentContent interaction.
	AddDoc	Function Name used in AddDoc interaction.
	DeleteDocs	Function Name used in DeleteDocs interaction.
	GetDocProperties	Function Name used in GetDocProperties interaction.
	UpdateDocProperties	Function Name used in UpdateDocProperties interaction.
	CancelDocPropertiesUpdate	Function Name used in CancelDocPropertiesUpdate interaction.
	FileDocsInFolder	Function Name used in FileDocsInFolder interaction.
	RemoveDocsFromFolder	Function Name used in RemoveDocsFromFolder interaction.
	GetDocFolders	Function Name used in GetDocFolders interaction.
	IsAnnotated	Function Name used in IsAnnotated interaction
	GetAnnotations	Function Name used in GetAnnotations interaction
	SaveAnnotations	Function Name used in SaveAnnotations interaction
Folder Interactions	GetFolderFolders	Function Name used in GetFolderFolders interaction.
	GetFolderAttributes	Function Name used in GetFolderAttributes interaction.
Queue Interactions	GetWorkspaces	Function Name used in GetWorkspaces interaction
	GetQueues	Function Name used in GetQueues interaction
	GetQueueFields	Function Name used in GetQueueFields interaction
	GetQueueEntries	Function Name used in GetQueueEntries interaction
	InsertQueueEntries	Function Name used in InsertQueueEntries interaction

Types Of Interactions	Function Name	Description
	DeleteQueueEntries	Function Name used in DeleteQueueEntries interaction
	UpdateQueueEntries	Function Name used in UpdateQueueEntries interaction
	CreateQueue	Function Name used in CreateQueue
	CreateWorkspace	Function Name used in CreateWorkspace interaction
Meta Data Interactions	GetDocClassIndices	Function Name used in GetDocClassIndices interaction.
	GetMenuValue	Function Name used in GetMenuValue interaction.
	GetSecurityInfo	Function Name used in GetSecurityInfo interaction
	GetDocClassDesc	Function Name used in GetDocClassDesc interaction
	GetMenuDesc	Function Name used in GetMenuDesc interaction
Password Status Interactions	GetPasswordStatus	Function Name used in GetPasswordStatus interaction
	ChangePassword	Function Name used in ChagePassword interaction
Print and fax Interactions	PrintDocs	Function Name used in PrintDocs interaction
	GetPrinterAttributes	Function Name used in GetPrinterAttributes interaction

Note: ISRA View edition only supports FindDocuments, GetDocumentContent, GetDocClassIndices, GetMenuValue, IsAnnotated, GetFolderAttributes, GetPasswordStatus, ChangePassword and, GetAnnotations functions.

- **InteractionVerb:** set to the default value of SYNC_SEND_RECEIVE defined in `InteractionSpec` interface always.
- **ExecutionTimeout:** the time in milliseconds (ms), that ISRA should wait to get a response from the IS. However, ISRA does not support this property.

The properties (in addition to those mentioned above) that apply to interactions whose output record is of type `javax.resource.cci.ResultSet` are:

- **FetchSize:** is an integer representing the number of rows that should be fetched from IS when more rows are needed for a `ResultSet`. If the user specifies the value as zero (0), CCI implementation of `ResultSet` uses a default value of 16.
- **FetchDirection**: valid value is FETCH_FORWARD**.**
- **ResultSetType**: valid value is TYPE_FORWARD_ONLY.
- **ResultSetConcurrency**: valid value is CONCUR_READ_ONLY.

Note: If values specified in the `InteractionSpec` object are invalid or not supported by ISRA, interaction is executed with default values, and appropriate Resource Warnings are

generated. These warnings can be retrieved using `getWarnings ()` method of Interaction.

The code snippet to create an `InteractionSpec` Object is:

```
import com.filenet.is.ra.cci.FN_IS_CciInteractionSpec;
FN_IS_CciInteractionSpec interactionSpec = new
FN_IS_CciInteractionSpec();
```

The getter and setter methods, as described in the Appendix A: References, are used to set the values for the properties.

Creating a Record Object

To create a Record object, invoke the `getRecordFactory()` method of the ConnectionFactory and create the record.

Getting a RecordFactory Instance

The `RecordFactory` creates `Record` instances just the way `ConnectionFactory` creates `Connection` instances. Invoke `getRecordFactory()` method of the `ConnectionFactory` to create the `RecordFactory` instance.

The code snippet to create a `RecordFactory` instance is:

```
RecordFactory recordFactory =
connectionFactory.getRecordFactory();
```

Creating a Record

After creating the `RecordFactory` instance, you can create:

- IndexedRecord.
- MappedRecord.

Creating an IndexedRecord

Invoke the `createIndexedRecord` method of `RecordFactory` to create an Indexed Record.

The code snippet to create an `IndexedRecord` is:

```
IndexedRecord iRecord =
recordFactory.createIndexedRecord("RecordName");
```

Creating a MappedRecord

Invoke the `createMappedRecord` method of `RecordFactory` to create a Mapped Record.

The code snippet to create a `MappedRecord` is:

```
MappedRecord mRecord =
recordFactory.createMappedRecord("RecordName");
```

Executing an Interaction

An Interaction can be executed by calling the `execute` method of an `Interaction` object. It has two variants:

- `public boolean execute(InteractionSpec interactionSpec, Record input, Record output)`

  This `execute` method takes an input record and updates the output record. The parameters accepted are:

  - `InteractionSpec: represents a target IS function.`
  - `Input Record: represents function specific input data.`
  - `Output Record: represents output data from the interaction.`

  It returns true, if the execution of the EIS function is successful and the output Record is updated. Otherwise, it returns false.

  The code sample to invoke an Interaction using this execute () method is:

```
InteractionSpec interactionSpec;
Record inputRecord, outputRecord;
boolean success = interaction.execute(interactionSpec,
inputRecord,
outputRecord);
```

- `public record execute(InteractionSpec interactionSpec, Record input).`

  This `execute` method takes an input record and returns an output record, if the execution of the Interaction is successful. It accepts these parameters:

  - `InteractionSpec: represents a target IS function.`
  - `Input Record: represents function specific input data.`

  This returns an output record if the execution of the IS function is successful. Otherwise, it returns `null`.

  The code sample to invoke an Interaction using this execute () method is:

```
InteractionSpec interactionSpec;
Record inputRecord;

Record outputRecord =
interaction.execute(interactionSpec,
inputRecord);
```

Note: If the output record is of type `javax.resource.cci.ResultSet`, then you can use only this variant of execute method.

Exception Handling

The execute () method of javax.resource.cci.Interaction throws javax.resource.ResourceException. A ResourceException provides:

- An ISRA specific string describing the error. This string is a standard Java exception message and is available through the `getMessage()` method.
- An ISRA specific error code that identifies the error condition represented by the ResourceException. The type of the error code returned is String and is available through `getErrorCode()` method.

In the next chapter, all the supported interactions by ISRA are documented in detail. At the end of each interaction, error codes with their descriptions are listed in a tabular form. Application components can handle the `ResourceException` and can take appropriate action.

3. Supported ISRA Interactions

ISRA is available in two editions:

- **View Edition:** This edition is for performing 'Read-Only' operations on IS. It supports interactions like searching of documents in an IS, retrieving the document content and document class indices details etc. from an IS server.
- **Enterprise Edition:** This edition is for 'Create/Update' operations on IS. Apart from supporting view edition interactions, this edition also supports interactions like committing and deleting documents, modifying document properties etc. in an IS.

The following table lists the interactions supported by each edition.

Document Interactions	
Function Name	**Supported Edition**
FindDocuments	View, Enterprise
GetDocumentContent	View, Enterprise
AddDoc	Enterprise
DeleteDocs	Enterprise
GetDocProperties	Enterprise
UpdateDocProperties	Enterprise
CancelDocPropertiesUpdate	Enterprise
FileDocsInFolder	Enterprise
RemoveDocsFromFolder	Enterprise
GetDocFolders	Enterprise
IsAnnotated	View, Enterprise
GetAnnotations	View, Enterprise
SaveAnnotations	Enterprise

Folder Interactions	
Function Name	**Supported Edition**
GetFolderFolders	Enterprise
GetFolderAttributes	View, Enterprise
Password Interactions	
Function Name	**Supported Edition**
GetPasswordStatus	View, Enterprise
ChangePassword	View, Enterprise
Print and Fax Interactions	
Function Name	**Supported Edition**
PrintDocs	Enterprise
GetPrinterAttributes	Enterprise

Queue Interactions		Meta Data Interactions	
Function Name	**Supported Edition**	**Function Name**	**Supported Edition**
GetWorkspaces	Enterprise	GetDocClassIndices	View, Enterprise
GetQueues	Enterprise	GetMenuValue	View, Enterprise
GetQueueFields	Enterprise	GetSecurityInfo	Enterprise
GetQueueEntries	Enterprise	GetDocClassDesc	Enterprise
InsertQueueEntries	Enterprise	GetMenuDesc	Enterprise
DeleteQueueEntries	Enterprise		
UpdateQueueEntries	Enterprise		
CreateQueue	Enterprise		
CreateWorkspace	Enterprise		

Note: All fields marked with * in the interaction description section below, are mandatory for an InputRecord.

Document Interactions

FindDocuments

FindDocuments interaction is used by the application component to retrieve data from all IS Document Index Records (DIRs), that match the input query statement.

To execute FindDocuments interaction, specify the input SQL query statement along with the maximum number of rows to be retrieved. A folder name can be specified to make the search more specific.

The input values for the FindDocuments interaction are specified through `QueryRequest` MappedRecord. The output values are returned through `QueryResult`, which is of type `ResultSet`. In the output, each row will contain the value for the index fields as specified in the SELECT clause of the query.

Note: FindDocuments only supports one variant of the `execute()` method, that takes `InteractionSpec` and `QueryRequest` record as parameters, and returns the output as `QueryResult` record.

The Input-Output Records for `FindDocuments` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
QueryRequest	MappedRecord	QueryResult	ResultSet	The output record contains columns that map to the SELECT clause of the input query. Each row contains value for the index fields as specified in the query.

Description of Input Record `QueryRequest`:

Key	Value Type	Remarks
query*	String	A SQL query statement.
folder_name	String	The search is performed in the specified folder. If the name is terminated with a backslash (/), then subfolders are also searched. This is an optional field.
max_rows*	Integer	The maximum number of rows to retrieve should always be greater than or equal to the InteractionSpec FetchSize property. This is a mandatory field.

Description of Output Record `QueryResult:`

The Output Record is of the type `javax.resource.cci.ResultSet`. The `ResultSet` contains columns that map to the SELECT clause of the input query statement. Each row contains the values for the index fields, as specified in the query.

Note: For the index field of type MENU, the menu label will be returned, instead of the menu value. You can use the returned menu label as an input to the GetMenuValue Interaction to get the menu value.

Query Specifications for FindDocuments

The SQL query denotes a 'SELECT' query statement and pertains to searching on the value of the index fields.

The code sample describes the supported SQL statement syntax, which an application component provides as '`query`' key to the `QueryRequest` MappedRecord input parameter of the `FindDocuments` interaction.

```
SELECT <selectable fields>

FROM <table name>

[WHERE <where clause>]

[KEY CONDITION <key condition>]
```

`<selectable fields>` represents the fields retrieved by the query. These fields define columns in the ResultSet.

Note: The wildcard selection, asterisk (*), is NOT supported. The column names in the SELECT, WHERE clause and in the KEY CONDITION are case sensitive.

<table name> is always FnDocument.

<where clause> is optional, it defines the query criteria and follows the syntax as described below:

<fieldcriteria> [<boolop> <fieldcriteria>]…where

<fieldcriteria> <field> <relop> <constant>|<field>

<boolop> AND, OR, NOT

<relop> <, <=, =, >, >=, !=, LIKE, IS NULL, IS NOT NULL

<field> can be an index field name, a number constant, or a string (including wildcard characters '?' and '*').

<key condition> defines an optional key criteria for the query and is limited to either of these two forms:

```
<keyfield> <relop2> <constant>

<keyfield> <relop2> <constant> AND <relop2> <constant>
```

where,

<keyfield> is a key index field defined on the IS server.

<relop2> <, <=, =, >, >=

Note: Parentheses can be used to control precedence.

These code samples are examples of the Select query statement:

```
SELECT F_DOCNUMBER, F_ENTRYDATE, PolicyID FROM FnDocument
WHERE F_ENTRYDATE > '8/1/1996' AND CustID IS NOT NULL

SELECT F_DOCNUMBER FROM FnDocument
WHERE F_PAGES > 1 KEY CONDITION F_DOCNUMBER < 123456
```

The IS system-defined fields and their acceptable usage in a query statement is listed in the table:

Field Name	Can be used in SELECT clause	Can be used in WHERE clause	IS Data Type	Description
F_ARCHIVEDATE	Yes	Yes	DATE	Archived date of the document.
F_CLOSED	Yes	No	BOOLEAN	True, if document is closed.
F_DELETEDATE	Yes	Yes	DATE	The date the document was/will be deleted.
F_DOCCLASSNAME	Yes	Yes	ASCII	Name of the document class.
F_DOCCLASSNUMBER	Yes	Yes	UNS_SHORT	ID of the document class.
F_DOCFORMAT	Yes	Yes	ASCII	Document's MIME-type and optional filename.
F_DOCLOCATION	Yes	Yes	ASCII	Description of external document location.

Field Name	Can be used in SELECT clause	Can be used in WHERE clause	IS Data Type	Description
F_DOCNUMBER	Yes	Yes	UNS_LONG	ID of the document.
F_DOCTYPE	Yes	Yes	BYTE	Indicates the document type.
F_ENTRYDATE	Yes	Yes	DATE	The date on which the document was committed.
F_PAGES	Yes	Yes	UNS_SHORT	Number of pages in the document.
F_RETENTBASE	Yes	No	BYTE	The date on which the retention period begins.
F_RETENTDISP	Yes	No	BYTE	Action to be taken once a document's retention period has ended.
F_RETENTOFFSET	Yes	Yes	UNS_LONG	Number of months from F_RETENTBASE before retention action is taken.

Note1: To find documents with only one page, use "F_PAGES IS NULL" in the WHERE clause, instead of specifying F_PAGES = 1.

Refer to Appendix A: References section for details on IS System-defined fields and corresponding Java data types.

Note2: The format for all Date fields supported by ISRA is 'DD/MM/YYYY hh:mm AM/PM'. If Date is mentioned in the format 'DD/MM/YYYY' then the default value of time set in the Date field is '00:00 AM'

The code sample shows how to find a Document with known Doc ID:

```
import java.sql.SQLException;


try {

// Define the type of interaction.

interactionSpec.setFunctionName ("FindDocuments");

//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord("QueryRequest");

//Insert values into the input record.

inputRecord.put ("query", "select F_DOCNUMBER,
F_DOCCLASSNAME, Account from FnDocument where F_DOCNUMBER
= 100022");

inputRecord.put("folder_name","myFolder");

inputRecord.put("max_rows", new Integer(4));
```

```
//Invoke the interaction with execute method that returns
the output record of type resultset.

ResultSet resultSet =(ResultSet)
interaction.execute(interactionSpec, inputRecord);

//Access each row in the resultset.

while (resultSet.next ()){
long docId = resultSet.getLong("F_DOCNUMBER");

String docClassName =
resultSet.getString(“F_DOCCLASSNAME”);

String account = resultSet.getString(“Account”);

System.out.println("DocId returned is:" + docId);

System.out.println("DocClassName returned is:" +
docClassName);
System.out.println("Account returned is:" + account);
}

}catch(ResourceException re){

      re.printStackTrace();

}catch(SQLException sqle){

      sqle.printStackTrace();

}
```

The error messages for `FindDocuments` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10001	Invalid column type.	An invalid data type is used to access data from the ResultSet.
FN_IS_RA_10002	Invalid column name.	A column name, which is not present in the table, is used.
FN_IS_RA_10003	ResultSet is closed.	If any of ResultSet method is called after the ResultSet is already closed.
FN_IS_RA_10004	Valid values are TYPE_FORWARD_ONLY for ResultSetType and FETCH_FORWARD for FetchDirection.	Fetch direction should be ‘FETCH_FORWARD’. ISRA supports ‘TYPE_FORWARD_ONLY’ ResultSet, and fetch direction should be ‘FETCH_FORWARD’.
FN_IS_RA_10005	Query generated no fields for ResultSet.	The ResultSet could not retrieve ‘column’ properties from the underlying EIS. This is an internal error.
FN_IS_RA_10006	Column Index out of range in ResultSet.	An invalid column index is used with any of getXXX () methods of the ResultSet.
FN_IS_RA_10007	FetchSize must be greater than zero .	FetchSize set on ResultSet is <= 0.

Error Code	Error Message	Description
FN_IS_RA_10008	Invalid cursor state.	This exception is thrown when ResultSet.next() is not called before accessing any of ResultSet methods.
FN_IS_RA_10009	End of ResultSet error.	If any of the ResultSet method is called after the last row is accessed from ResultSet.
FN_IS_RA_10010	Error occurred while generating ResultSetMetaData.	The ResultSet could not generate metadata information (number, types and properties of this ResultSet object's columns).
FN_IS_RA_10011	Clone is not supported.	If clone() method is called on the ResultSet (ISRA does not support ResultSet cloning).
FN_IS_RA_10012	getXXX() is not supported in this version of ISRA.	A method is called, which ISRA ResultSet does not support.
FN_IS_RA_10013	updateXXX methods are not supported in this version of IS RA.	An updateXXX method is called (updateXXX methods are not supported on ISRA ResultSet).
FN_IS_RA_10367	Query cannot be null in interaction FindDocuments.	SQL query cannot be null as it is a mandatory parameter in FindDocuments interaction.
FN_IS_RA_10368	The key 'max_rows' cannot be less than one (1) in interaction FindDocuments.	The Max rows parameter as specified should be greater than or equal to one and is a mandatory value.
FN_IS_RA_10370	<90,0,53>Invalid folder name.	The correct syntax for the folder name is “/Account”. Make sure that the folder name contains a ‘/’ in the beginning.
FN_IS_RA_10370	<90,0,9>No folder with name and state specified exist.	The folder name specified does not exist on the IS, or the user does not have access to the specified folder. Check for the correct folder name.
FN_IS_RA_10370	The folder length exceeds 152 characters.	The folder name can contain a maximum of 152 characters, including the preceeding ‘/’ character.
FN_IS_RA_10370	A sub-folder length exceeds 18 characters.	A sub-folder length is limited to a maximum of 18 characters including the slash.
FN_IS_RA_10370	<90,0,49> Attempt to create too many folder levels.	There can be maximum of 8 folder levels, each separated by a slash.
FN_IS_RA_10380	The key 'max_rows' cannot be null in interaction FindDocuments.	The Max rows parameter should be greater than or equal to one, and is a mandatory value.
FN_IS_RA_10388	Max Rows should be an Integer in interaction FindDocuments.	Max. Rows object should be of type `java.lang.Integer` object in FindDocuments interaction.
FN_IS_RA_10367	Invalid Query in Interaction ‘Find Documents’	The SQL query specified is not syntactically correct. See section ‘Query Specifications for FindDocuments’ for the specifications of SQL query.

Error Code	Error Message	Description
FN_IS_RA_10854	Specified key field in Key Condition does not exist.	The specified key field in Key Condition does not exist on the IS.
FN_IS_RA_10856	Invalid Filter Condition.	This exception is thrown when filter condition in the query is invalid. See section 'Query Specifications for FindDocuments' for the specifications of SQL query.
FN_IS_RA_10857	'*' is not supported in the SELECT clause.	This exception is thrown when '*' is present with the SELECT clause of the query.
FN_IS_RA_10858	Missing space or parenthesis in query.	If there is no white space or a parenthesis after a closing single quote, the exception is thrown.
FN_IS_RA_10859	Invalid operator in query.	Invalid operator in query.
FN_IS_RA_10860	Missing right quote in query.	Missing right quote in query.

GetDocumentContent

GetDocumentContent interaction is used by the application component to retrieve the requested Document's page content.

The user has to specify the Document-ID and the page number of the document to be retrieved.

The GetDocument interaction results in the migration of the physical page from the optical disk to the default cache or the specified cache.

The pre-fetch count can be specified to pre-fetch the number of additional pages from the optical disk.

The polling interval is the frequency (in seconds) at which ISRA checks to determine if the document has been migrated to cache or not. The valid value for polling interval is between 1-5 seconds. The default value is 5 seconds. The polling interval can also be specified in milliseconds for this interaction. The valid value for polling interval (in ms) is in between 250-5000 milliseconds. The default value is 5000 milliseconds.

The Record `DocContentRequest`, of the type `MappedRecord`, is used to specify the input values. The content of the requested page is returned as a stream of bytes contained in the Record `DocContent`, which is of type `MappedRecord`.

The Input-Output Records for `GetDocumentContent` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocContentRequest	MappedRecord	DocContent	MappedRecord	The output record contains an object of type InputStream. This class provides methods to read the stream of bytes (as binary data).

Description of the Input Record `DocContentRequest`:

Key	Value Type	Remarks
doc_id*	Long	The Document-ID. It is a mandatory field.
page_number*	Integer	The page number of the document. It is a mandatory field.
page_cache	String	Name of the IS page cache to migrate the page to. If not specified the default cache is used. It is an optional value. Page cache name should be in <object name>: <domain name>: <organization name> format.
prefetch_count	Integer	Number of additional pages to pre-fetch from the optical disk, to increase the relative access speed of the document. It is an optional value. Default value is 0.
polling_interval	Integer	The polling interval used to see if the document is migrated from optical storage. Default value is 5.
polling_interval_ms	Integer	The polling interval used to see if the document is migrated from optical storage. Valid value is in between 250-5000.Default value is 5000.
no_of_times_to_poll	Integer	Number of times to poll. Default value is 1.
checksum_enabled	Boolean	Checksumming to be done for the document being retrieved. Default value is false.

Description of Output Record `DocContent`:

Key	Value Type	Remarks
stream	InputStream	It contains the page data.
mime_type	String	MIME/content type of the returned content.
file_name	String	File name associated with the document.

Note: For documents that are NOT present in the cache, the application component might get an object does not exist error, <77,0,10>. In such a scenario, the application component would re-issue the GetDocumentContent request, after waiting for some duration. In ISRA3.0a, a new parameter, *no_of_times_to_poll*, has been added. ISRA polls for n number of times (where n is *no_of_times_to_poll*) where each polling time is t secs (where t is *polling_interval*).

The points to consider when you execute the GetDocument interaction are:

- While requesting the retrieval of a particular page of a document, the Document-ID and the page number of the document must be specified through the keys ‘doc_id’ and ‘page_number’. This should be done using the input MappedRecord `DocContentRequest`.
- A specific `page_cache` can be specified to migrate the pages, by using the ‘page_cache’ key of the `DocContentRequest`. By specifying the cache name, you may have faster access to the page. This is because the page whose content you want to read may already reside in this page_cache. Thus, you do not have to wait for the page to migrate from the optical disk to the cache.

- If both polling_interval and polling_interval_ms are specified then polling_interval will take precedence over polling_ interval_ms.

Note: If the page cache is not specified, the document will be migrated to the default cache of the domain to which the user is logged in.

- You can also specify the number of documents to be pre-fetched from the optical disk, using the key 'prefetch_count'. This assumes that you want to access the pages that follow the page number specified in the key 'page_number'. Executing this interaction will give you the total number (1 + prefetch_count) of pages, to be brought into the cache. This speeds up the access to the subsequent pages.

The code sample to get document content is:

```
import java.io.*;


try{
interactionSpec.setFunctionName("GetDocumentContent");
MappedRecord inputRecord =
recordFactory.createMappedRecord ("DocContentRequest");
//Insert values into the input record.
inputRecord.put("doc_id", new Long(200123));
inputRecord.put("page_number", new Integer(1));
inputRecord.put("page_cache","page_cache1:FNIS:FileNet");
inputRecord.put("prefetch_count", new Integer(4));
inputRecord.put("polling_interval", new Integer(4));
inputRecord.put("polling_interval_ms", new
Integer(4000));
inputRecord.put("no_of_times_to_poll", new Integer(2));
inputRecord.put("checksum_enabled", new Boolean(true));
/*Invoke the interaction with execute method that returns
the output record of type mapped record.*/
MappedRecord outputRecord = (MappedRecord)
interaction.execute(interactionSpec, inputRecord);
InputStream inputStream =
(InputStream)outputRecord.get("stream");
String fileName = (String) outputRecord.get("file_name");
String mimeType = (String) outputRecord.get("mime_type");
int availableBytes = inputStream.available();
byte [] data = new byte[availableBytes];
inputStream.read(data);
```

```
System.out.println("Mime type of the returned content:" +
mimeType);

System.out.println("Name of the file:" + fileName);

System.out.println("Size of the returned content:" +
data.length + " bytes");

catch (ResourceException e) {

      e.printStackTrace();

}catch(IOException ioe){

        ioe.printStackTrace();

}
```

The error messages for `GetDocumentContent` Interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10360	The key 'doc_id' cannot be null in interaction GetDocumentContent.	Doc id cannot be null in interaction GetDocumentContent.
FN_IS_RA_10361	Domain name and/or Organization name in the key 'page_cache' is not valid.	The page cache name as specified should belong to the same domain where you are logged on. No cross-domain cache names are allowed. For example. if you are logged on to 'DomainA:OrganizationA', then a valid cache name could be 'page_cache1: DomainA: OrganizationA', not 'page_cache1:DomainB:OrganizationB'.
FN_IS_RA_10362	TimeOut cannot be less than polling_interval.	The TimeOut value cannot be less than polling interval.
FN_IS_RA_10363	<80,0,2> Document not found by DOC.	The DocID passed, as the input parameter either does not exist on the IS server or the user does not have permissions to view the document.
FN_IS_RA_10363	<80,0,18> Page number out of the range of pages in a document given to DOC.	The requested page number does not exist. Enter a valid value for the page number you want to view. By default, each document has at least one page.
FN_IS_RA_10363	Invalid cache name specified.	The specified cache does not exist. Check the correct name from your IS administrator. For example, page_cache1: DomainA:OrganizationA, where DomainA: OrganizationA, refers to the domain to which you are logged on.
FN_IS_RA_10381	The key 'page_number' cannot be null in interaction FindDocuments.	The page number cannot be zero or null.
FN_IS_RA_10382.	The key 'page_number' cannot be less than one (1) in interaction GetDocumentContent.	The page number cannot be zero or an empty value.
FN_IS_RA_10383	The key 'prefetch_count' cannot be less than zero (0) in interaction	Specified value should be greater than zero.

Error Code	Error Message	Description
	GetDocumentContent.	
FN_IS_RA_10384	The key 'polling_interval' should be greater than than 250 milliseconds and less than equal to five (5) seconds.	Specify a valid polling interval (in seconds) value in the range (1, 5) (inclusive of 1 and 5).
FN_IS_RA_10385	The key 'doc_id' cannot be less than one (1) in interaction GetDocumentContent.	Specified value should be greater than zero.
FN_IS_RA_10389	Doc Id object should be object of type Long in interaction GetDocumentContent.	DocId should be object of type Long in interaction GetDocumentContent.
FN_IS_RA_10390	Page Number should be an Integer in interaction GetDocumentContent.	Page Number should be an Integer.
FN_IS_RA_10391	Polling Interval should be an Integer in interaction GetDocumentContent.	Polling Interval should be an Integer.
FN_IS_RA_10392	Prefetch Count should be an Integer in interaction GetDocumentContent.	Prefetch Count should be an Integer.
FN_IS_RA_11071	IS calculated Checksum value and Local Checksum value is not identical. Data might be corrupt!	The checksum value stored in the IS for the particular document does not match the checksum value calculated by ISRA. This error signifies that there is a high probability of the document data being corrupt.

AddDoc

The AddDoc interaction is used by the application component to add a document to the IS server. The application component can specify a destination folder and migration options for document archival.

With the AddDoc interaction the application component specifies the document class name, document type, document family name, document index record, read-write-execute permissions, etc. The pages to be added are given as an array of `java.io.InputStream`. Each element of the array represents one page in the document. After the document is added, it can be filed in folders specified by the client, in the Record `FolderSet`.

All these values are specified through the `AddDocRequest` MappedRecord. ISRA returns the document ID of the added document in the MappedRecord, `DocID`.

The Input-Output Records for `AddDoc` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
AddDocRequest	MappedRecord	DocID	MappedRecord	The output record will contain the doc id of the added document.

Description of Input Record `AddDocRequest`:

Key	Value Type	Description	Default Value
doc_class_name*	String	Name of the document's class.	
doc_type	Short	Type of document, for example IMAGE, TEXT etc. Refer to AddDoc and UpdateDoc Interactions.	53
doc_family_name	String	Family name for the media surface.	null
is_duplication_okay	Boolean	If the value is false and the document is a duplicate of an existing document, then the document will not be added to IS.	False
cluster_key	Object	cluster_key is not supported in this release, and should not be set.	Exception is thrown if a value is set.
sec_read	Long	Read access attribute.	1
sec_write	Long	Write access attribute.	1
sec_append_execute	Long	Append or execute access attribute.	1
cache_name	String	Name of an IS page cache.	null
index_records*	DocProperties	IndexedRecord of `DocProperty` records for the document that is being added.	
doc_page_streams*	InputStream[]	One stream for each document page. **Note**: The array doc_page_streams should not contain the same InputStream object reference more than once.	
folder_set	FolderSet	Folders in which to file this document.	Null
checksum_enabled	Boolean	Checksumming to be done for the document to be added	False

Description of the Output Record `DocID`:

Key	Value Type	Description
doc_id	Long	Returned Document ID.

Note: Refer to Appendix section for more details on this interaction.

The code sample to add a document is:

```
import java.io.InputStream;
import java.io.FileInputStream;
```

```
import java.io.*;


try{
// Define the type of interaction.
interactionSpec.setFunctionName ("AddDoc");
//Create the input mapped record.
MappedRecord inputRecord=
recordFactory.createMappedRecord(“AddDocRequest");
IndexedRecord collectionDocProperties =
recordFactory.createIndexedRecord("DocProperties");
IndexedRecord folderSet =
recordFactory.createIndexedRecord("FolderSet") ;
      //create an array of inputstreams
 InputStream[] pageStreams = new InputStream[2];
      /* Create an array of InputStream from which the
document data can be read by ISRA */


//insert values into the folder set.
    folderSet.add(new String("folder_name"));
//Insert values into the input record.
inputRecord.put ("doc_class_name", "Account");
inputRecord.put ("index_records",
collectionDocProperties);
inputRecord.put ("folder_set", folderSet);
inputRecord.put ("doc_page_streams", pageStreams);
inputRecord.put ("doc_type", new Short("49"));
inputRecord.put ("doc_family_name", "OpticalFamily1");
inputRecord.put ("is_duplication_ok", new Boolean(true));
inputRecord.put ("sec_read", new Long(0));
inputRecord.put ("sec_write", new Long(0));
inputRecord.put ("sec_append_execute", new Long(0));
inputRecord.put ("cache_name",
"page_cache1:FNIS:FileNet");
inputRecord.put ("checksum_enabled", new Boolean(true));
```

```
/*Invoke the interaction with execute method that returns
the output record of type MappedRecord. */

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

//Access new Doc ID from the mapped record.

long resultDocID = new Long((
resultMappedRecord.get("doc_id")
).toString()).longValue();

System.out.println("Document ID for added record is:" +
resultDocID);

}catch(ResourceException e){

      e.printStackTrace();

}
```

The error messages for AddDoc interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is not available in ISRA View Edition.
FN_IS_RA_10401	Invalid document class name.	The document class is either null, or not an object of type `java.lang.String` or blank String.
FN_IS_RA_10402	Invalid document type object.	Specified document type object is not of type `java.lang.Short`.
FN_IS_RA_10403	cluster_key is not supported.	Cluster key is not supported in AddDoc interaction.
FN_IS_RA_10404	Invalid security read attribute object.	Specified security read object is not of type `java.lang.Long.`
FN_IS_RA_10405	Invalid security write attribute object.	Specified security write object is not of type `java.lang.Long.`
FN_IS_RA_10406	Invalid security append or execute attribute object.	Specified security append-execute object is not of type `java.lang.Long.`
FN_IS_RA_10407	Invalid family name.	Specified family name is not of type `java.lang.String`.
FN_IS_RA_10408	Invalid isDuplicationOk object.	Specified isDuplicationOk object is not of type `java.lang.Boolean.`
FN_IS_RA_10409	Invalid cache name.	Specified cache name is not of type `java.lang.String.`
FN_IS_RA_10411	Invalid document streams.	Specified InputStream[] is null or blank.
FN_IS_RA_10412	Invalid folder set object type.	Specified folder set object is not valid (folder set object should be of type javax.resource.cci.IndexedRecord).
FN_IS_RA_10414	Invalid folder name.	Specified folder name object is not valid (folder name object should be of type

Error Code	Error Message	Description
		java.lang.String).
FN_IS_RA_10416	<80,0,16>No matches found when attempting to find information from DOC.	The doc family name provided is invalid.
FN_IS_RA_10423	Invalid InputStream object.	Specified InputStream object inside the InputStream[] is null.
FN_IS_RA_10429	Invalid Checksum object.	Invalid object passed for checksum value.

DeleteDocs

The DeleteDocs interaction is used by the application component to delete the specified documents from the IS server. The input record is a List of 'doc_id' corresponding to the documents to be deleted. The output `DocErrorSet` IndexedRecord contains the `DocError` MappedRecord for each document whose delete failed. `DocError` contains the doc Id and the corresponding error code.

The Input-Output Records for `DeleteDocs` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocSet	IndexedRecord	DocErrorSet	IndexedRecord	DocErrorSet contain DocError MappedRecords for those documents, which could not be deleted.

Description of Input Record DocSet:

This is an IndexedRecord of `doc_id` items. `doc_id` is of type `Long`.

Description of the Output Record DocErrorSet:

This is an IndexedRecord of `DocError` MappedRecords.

The `DocError` MappedRecord is described in the following table:

Key	Value Type	Description
doc_id	Long	The document id that is not deleted.
error_code	Long	IS error code for the above document id.

Note: ISRA does not return any error for invalid document id(s) in the DeleteDocs interaction. If this interaction is performed by a user who has Read and Write permissions on a document(s) but does not have Append/Execute permission, the document(s) may be left in an inconsistent state. ISRA will return a valid error message but it would have deleted the document from the Index database, making it inaccessible. However, such a document can be successfully deleted (by retrying) by performing this operation with a User with higher (proper) security credentials.

The code sample to delete documents is:

```
import java.util.Iterator;
import java.util.Map;
import java.util.HashMap;


try {
// Define the type of interaction.
interactionSpec.setFunctionName("DeleteDocs");
//Create the input Indexed record.
IndexedRecord inputRecord=
recordFactory.createIndexedRecord("DocSet");
//Insert values into the input record.

//docSet - it is an IndexedRecord containing list of doc
ids.
inputRecord.add(new Long(123456));
inputRecord.add(new Long(123459));
/*Invoke the interaction with execute method that returns
an output record of type IndexedRecord contains list of
DocError MappedRecords.*/
IndexedRecord resultIndexedRecord =
recordFactory.createIndexedRecord("DocErrorSet");
resultIndexedRecord =(IndexedRecord)
interaction.execute(interactionSpec, inputRecord);
//Create Iterator for Indexed Record.
Iterator iterator = resultIndexedRecord.iterator();
while(iterator.hasNext()){
Map mapResult = new HashMap();
mapResult.putAll((Map)iterator.next());
System.out.println("Doc ID:"+mapResult.get("doc_id"))
System.out.println("ErrorCode:"+mapResult.get("error_code
"));
}
System.out.println("Document(s) deleted");
}catch(ResourceException re){
      re.printStackTrace();
}
```

The error messages for DeleteDocs interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is not available in ISRA View Edition.
FN_IS_RA_10413	Invalid document Id object type.	Document id object is either null or not a Long object.
FN_IS_RA_10415	Invalid DocSet Record.	The DocSet index record is empty, i.e., does not contain any doc ids to be deleted.

GetDocProperties

GetDocProperties is used by the application component to retrieve the properties of a document from an IS server. The application component needs to specify the Doc ID and the flag 'is_lock_desired'.

If it is not required to update the document properties, an update lock should not be requested and the key 'is_lock_desired' in the input record DocPropertiesRequest should be set to false. If the key 'is_lock_desired' is set to 'true', the document properties will be locked for updating and a capability structure will be returned. Capability structure is an array of long of length 4.

The input values are DocumentPropertiesRequested MappedRecord and, the output values are returned in a DocumentPropertiesReturned MappedRecord.

The Input-Output Records for GetDocProperties are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocumentProperties Request	MappedRecord	DocumentProperties Return	MappedRecord	-

Description of the Input Record DocumentPropertiesRequest:

Key	Value Type	Description	Default Value
doc_id*	Long	Document ID of the document whose properties are to be retrieved.	-
is_lock_desired	Boolean	Specifies if it is required to lock the properties for update.	False

Description of the Output Record DocumentPropertiesReturn:

Key	Value Type	Description
doc_properties	DocProperties	DocProperties is an IndexedRecord of DocProperty MappedRecord objects.
item_lock	long[4]	Capability structure.

Description of MappedRecord `DocProperty`:

Key	Value Type	Description
index_name	String	Index name represented as a String
index_type	Short	Index type represented as a Short.
index_value	Object	The exact type of value depends upon index type as per the DocClass definition which can be retrieved using GetDocClassIndices interaction.

The code sample to get the document properties is:

```
import java.util.*;


try{

// Define the type of interaction.

interactionSpec.setFunctionName ("GetDocProperties");

//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord("DocumentPropertiesReque
st");

//Insert values into the input record.

inputRecord.put ("doc_id",new Long(10008));

inputRecord.put ("is_lock_desired", new Boolean(true));

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord, containing
DocProperties IndexedRecord, item lock(capability
structure.)*/

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

/*Get IndexedRecord of doc properties from
resultMappedRecord.*/

IndexedRecord indexedRecord = (IndexedRecord)
resultMappedRecord.get("doc_properties");

Iterator iterator = indexedRecord.iterator();

Map map;

while(iterator.hasNext()){

map = new HashMap();

map.putAll((Map)iterator.next());

System.out.println("Index Name:” +
map.get(“index_name”));
```

```
}
long[] capability = new long[4];
//Get capability structure from resultMappedRecord.
capability = (long[])resultMappedRecord.get("item_lock");
for(int j=0;j<4;j++)
System.out.println("Item Lock" + j + "=" + capability[j]
);
}
catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `GetDocProperties` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10413	Invalid document Id object type.	Document ID object is either null or not `Long` object.
FN_IS_RA_10451	Invalid isLockDesired object type.	IsLockDesired object is not a Boolean object.
FN_IS_RA_10458	<90,0,6>Requested record not found.	The specified DocID is not present.
FN_IS_RA_10458	<92,2,5>Read permission is denied.	The user does not have Read permission on the specified document to view the properties.
FN_IS_RA_10458	<92,2,6>Write permission is denied.	The user does not have Write permission to update the properties of a document.

UpdateDocProperties

UpdateDocProperties is used by the application component to update the properties of a document. To update the document properties, the application component has to first get the properties using the `GetDocProperties` interaction, with the key 'is_`lock`_desired' set to '`true`'. This will lock the document properties for update and return a capability structure, which is an array of `long` of length 4.

The Interaction object used for `UpdateDocProperties` should be same as the one used for the preceding `GetDocProperties`. The Interaction object used in `GetDocProperties` interaction stores the Document ID and the capability structure returned from the IS.

The Input-Output Records for `UpdateDocProperties` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocumentProperties Return	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `DocumentPropertiesReturn`:

Key	Value Type	Description	Default Value
doc_properties*	DocProperties	Properties to update	-
item_lock*	long[4]	Long array of length 4. It is the capability structure, which was returned from a preceding call to GetDocProperties,	-

Description of the Output Record `GenericResult`:

Key	Value Type	Description
Result	Long	Zero if the interaction is successful. Otherwise, it contains the IS error tuple.

Note: Refer to Appendix section setInherentLogin()

Public void setInherentLogin(Integer inherentLogin)

Sets the Inherent method that ISRA will use for all logins unless specified by the user.

setDeploymentInstance()

public void setDeploymentInstance (Integer deploymentInstance)

Sets the value of the instance of ISRA that is being deployed on a single machine.

AddDoc and UpdateDocProperties Interactions for more details on input parameters to this interaction.

The code sample to update doc properties is:

```
try{

// Define the type of interaction.

interactionSpec.setFunctionName("UpdateDocProperties");

//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord("DocumentPropertiesRetur
n");

//Insert values into the input record.
```

```
/* docProperties IndexedRecord contains document property
MappedRecords.*/

IndexedRecord docProperties =
recordFactory.createIndexedRecord( "DocProperties");


      MappedRecord docPropMappedRecord =
recordFactory.createMappedRecord("DocProperty");

      docPropMappedRecord.put("index_name",
"test_index_name");

      docPropMappedRecord.put("index_value",
"test_value");

      docPropMappedRecord.put( "index_type", new
Short("50") );

      docPropMappedRecord.put("index_name",
"F_DOCNUMBER");

      docPropMappedRecord.put("index_value", new
Long(100016));

      docPropMappedRecord.put( "index_type", new
Short("71") );


inputRecord.put("doc_properties", docProperties);

//Capability is array of long which is obtained after
executing 'GetDocProperties' interaction.

long[] capability = new long[4];

inputRecord.put("item_lock", capability);

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord contains result of
the interaction i.e. success or failure code.*/

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

//Get result from Mappedrecord.

//result=0: Document Properties are updated successfully.

/*result!=0 : Error in updating Document Properties, and
error id = result. */

long result = new
Long(resultMappedRecord.get("result").toString()).longVal
ue();

if (result==0)
```

```
System.out.println("Update of Document Properties is done
successfully.");
else
System.out.println("Error " + result + " has occurred in
updating the Document properties.");
}
catch (Exception e) {
      e.printStackTrace();
}
```

The error messages for the `UpdateDocProperties` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10400	F_DOCNUMBER property cannot be null.	`F_DOCNUMBER` property cannot be null in interaction UpdateDocProperties.
FN_IS_RA_10416	No matches found when attempting to find information from DOC.	The doc family name provided is invalid.
FN_IS_RA_10452	Invalid DocProperties Record object.	`DocumentProperties` object is either null or not a `javax.resource.cci.IndexedRecord` object.
FN_IS_RA_10453	Invalid DocProperty Record object.	`DocumentProperty` object is either null or not a `javax.resource.cci.MappedRecord` object.
FN_IS_RA_10454	Invalid capability object.	Capability structure object is either null or not a `long [].`
FN_IS_RA_10455	Invalid document Id.	The document ID is not same as the document ID whose properties had been retrieved and locked using GetDocProperties interaction.
FN_IS_RA_10458	<90,0,6>Requested record not found.	The requested record is not present in the IS.
FN_IS_RA_10458	<92,2,5>Read permission is denied.	The user does not have Read permission on the specified document to view the properties.
FN_IS_RA_10458	<92,2,6>Write permission is denied.	The user does not have Write permission on specified document to update the properties.

CancelDocPropertiesUpdate

`CancelDocPropertiesUpdate` is used by the application component to cancel a pending update and release the lock acquired on the document properties.

The input record DocPropertiesReturn should contain the DocProperties IndexedRecord, and the item lock (capability structure), acquired using GetDocProperties interaction.

In the DocProperties IndexedRecord, all document property values can be null except the system index property whose index name is 'F_DOCNUMBER'. All other index property records (DocProperty MappedRecords) are ignored if passed by application component.

The Input-Output Records for CancelDocPropertiesUpdate are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocumentProperties Return	MappedRecord	GenericResult	MappedRecord	The output record will contain the result of the interaction, i.e., success or failure.

Description of the Input Record DocumentPropertiesReturn:

Key	Value Type	Description	Default Value
doc_properties*	DocProperties	IndexedRecord of DocProperty MappedRecords.	-
item_lock*	long[4]	IS capability structure.	-

Description of the Output Record GenericResult:

Key	Value Type	Description
Result	Long	Returns an error tuple on failure.

The code sample to cancel document properties update is:

```
try  {

// Define the type of interaction.

interactionSpec.setFunctionName("CancelDocPropertiesUpdat
e");

//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord("DocumentPropertiesRetur
n");

//Insert values into the input record.

// docProperties IndexedRecord contains document property
MappedRecords.

IndexedRecord docProperties =
recordFactory.createIndexedRecord("DocProperties");
```

```
MappedRecord docProperty =
recordFactory.createMappedRecord("DocProperty");

docProperty.put(“index_name”, “F_DOCNUMBER”);

docProperty.put(“index_type”, new Short(“71”));

docProperty.put(“index_value”, new Long(123456));

docProperties.add(docProperty);

inputRecord.put("doc_properties", docProperties);

//Capability is array of long which is obtained after
executing ‘GetDocProperties’ interaction.

long[] capability = new long[4];

inputRecord.put("item_lock", capability);

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord contains result of
the interaction i.e. success or failure code.*/

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

//Get result from Mappedrecord.

//result=0 : Update has been cancelled.

/*result!=0 : Error in canceling Property Update, error
id = result. */

long result = new
Long(resultMappedRecord.get("result").toString()).longVal
ue();

if (result==0)

System.out.println(“Update Properties are cancelled
successfully.”);

else

System.out.println(“Error “ + result + “ has occurred in
canceling the update.”);

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for the `CancelDocPropertiesUpdate` interaction are listed in the following table:

Error Code	Error Message	Description

FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10452	Invalid DocProperties Record object.	DocumentProperties object is either null or not `javax.resource.cci.IndexedRecord` object.
FN_IS_RA_10453	Invalid DocProperty Record object.	DocumentProperty object is either null or not `javax.resource.cci.MappedRecord` object.

FileDocsInFolder

FileDocsInFolder is used, by the application component, to assign documents to a folder. The input record for this interaction is `DocumentsAndFolder` MappedRecord. This record contains the name of the destination folder, a `DocSet` that is a set of document IDs and a parameter `place_after`, which is a doc ID after which the documents are to be filed. To put the document at the beginning of a folder, use a value of zero (0) for `place_after`, and to put it at the end of the folder, use either the document id of the last document in the folder or the value 4294967295.

The Input-Output Records for `FileDocsInFolder` are explained in the following table:

Input Record Description		**Output Record Description**		**Remarks**
Name	**Type**	**Name**	**Type**	
DocumentsAndFolder	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `DocumentsAndFolder`:

Key	**Value Type**	**Description**	**Default Value**
destination_folder*	String	Destination folder.	-
documents*	DocSet	List of doc_ids to be added.	-
place_after	DocID	Doc id after which the documents are to be added.	-

Description of the Output Record `GenericResult`:

Key	**Value Type**	**Description**
result	Long	Result code, the IS error tuple.

The code sample to assign the specified documents to a specific folder is:

```
try {
// Define the type of interaction.
interactionSpec.setFunctionName("FileDocsInFolder");
//Create the input mapped record.
MappedRecord inputRecord=
recordFactory.createMappedRecord("DocumentsAndFolder");
```

```
//Inserte values into the input record.

inputRecord.put("destination_folder", "folder1");

//docSet is an IndexedRecord containing list of docIds.

IndexedRecord docSet =
recordFactory.createIndexedRecord("DocSet");

docSet.add(new Long(123456));

//docSet.add(new Long(112233));

inputRecord.put("documents", docSet);

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord contains the
result success or failure code. */

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

//Get result value as long resultMappedRecord.

long resultErrorID = new Long((resultMappedRecord.get
("result")).toString()).longValue();

//Returns the error id.

if (resultErrorID!=0)

System.out.println("Error Code : " + resultErrorID);

else

System.out.println("Document(s) filed successfully");

}catch(Exception e){

    e.printStackTrace();

}
```

The error messages for `FileDocsInFolder` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10413	Invalid document Id object type.	Document id is either null or not a `Long` object.
FN_IS_RA_10414	Invalid folder name.	Folder name is either null or not a `String` or blank string.
FN_IS_RA_10415	Invalid DocSet Record.	DocSet object is either null or not a `javax.resource.cci.IndexedRecord`.

RemoveDocsFromFolder

RemoveDocsFromFolder is used by the application component to remove documents from a folder. The input record for this interaction is a `DocumentsAndFolder` MappedRecord. This record contains the name of a folder and `DocSet`. `DocSet` is a set of documents to be removed from the specified folder.

The Input-Output Records for `RemoveDocsFromFolder` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocumentsAndFolder	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `DocumentsAndFolder`:

Key	Value Type	Description	Default Value
destination_folder*	String	Destination folder.	-
documents*	DocSet	List of Documents to be removed	-
place_after	DocID	Will always be null.	-

Description of the Output Record `GenericResult`:

Key	Value Type	Description
Result	Long	Result code, the IS error tuple.

The code sample to remove docs from folder:

```
try {
// Define the type of interaction.
interactionSpec.setFunctionName ("RemoveDocsFromFolder");
//Create the input mapped record.
MappedRecord inputRecord=
recordFactory.createMappedRecord("DocumentsAndFolder");
//Insert values into the input record.
inputRecord.put("destination_folder", "folder1");
//docSet is an IndexedRecord containing list of docIds.
IndexedRecord docSet =
recordFactory.createIndexedRecord("DocSet");
docSet.add(new Long(123456));
docSet.add(new Long(112233));
inputRecord.put("documents", docSet);
```

```
/*Place_after parameter is always null for this
interaction.*/

inputRecord.put("place_after", null);

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord contains the
result success or failure.*/

MappedRecord resultMappedRecord =(MappedRecord)
interaction.execute(interactionSpec, inputRecord);

//Get result value as long resultMappedRecord.

long resultErrorID = new
Long((resultMappedRecord.get(“result”)).toString()).longV
alue();

//Returns the error id.

if (resultErrorID!=0)

System.out.println("Error Code : " + resultErrorID);

else

System.out.println("RemoveDocsFromFolder completed
successfully.......");

}catch(ResourceException e){

      e.printStackTrace();

}
```

The error messages for `RemoveDocsFromFolder` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10413	Invalid document Id object type.	Document id is either null or not a `Long` object.
FN_IS_RA_10414	Invalid folder name.	Folder name is either null or not a `String` or blank string.
FN_IS_RA_10415	Invalid DocSet Record.	DocSet object is either null or not a `javax.resource.cci.IndexedRecord`.
FN_IS_RA_10418	The folder length exceeds 152 characters.	The folder name can contain a maximum of 152 characters, including the preceeding ‘/’ character.
FN_IS_RA_10418	A sub-folder length exceeds 18 characters.	A sub-folder length is limited to a maximum of 18 characters including the slash.
FN_IS_RA_10418	<90,0,49> Attempt to create too many folder levels.	There can be maximum of 8 folder levels, each separated by a slash.

GetDocFolders

GetDocFolders is used by the application component to retrieve the set of folders that has the filed document. The input record, `DocID` contains the `doc_id` of the document for which you want to find where the document is filed (the folders). The output record `FolderSet` is a list of folder names.

The Input-Output Records for `GetDocFolders` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocID	MappedRecord	FolderSet	IndexedRecord	-

Description of the Input Record `DocID`

Key	Value Type	Description	Default Value
doc_id*	Long	The doc_id whose folders need to be retrieved.	-

Description of the Output Record `FolderSet`

FolderSet is an IndexedRecord that contains folder names as String objects.

The code sample to retrieve the set of folders in which the specified document is filed is:

```
try {

// Define the type of interaction.

interactionSpec.setFunctionName("GetDocFolders");

//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord(“DocID");

//Insert values into the input record.

inputRecord.put("doc_id",new Long(10006));

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord containing list of
Maps.*/

IndexedRecord folderSet =(IndexedRecord)
interaction.execute(interactionSpec, inputRecord);

for(int i = 0; i < folderSet.size(); i++)

{

System.out.println(“Folder name is: ”+ folderSet.get(i));

}

}catch (ResourceException e) {

      e.printStackTrace();
```

```
}
```

The error messages for `GetDocsFolder` interaction are listed in the following table:

Table 1: GetDocsFolder – Error Messages

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10413	Invalid document Id object type.	Document id is either null or not a Long object.
FN_IS_RA_10419	<90,0,6>Requested record not found.	The specified Doc Id is not present.

IsAnnotated

The IsAnnotated is used by the application component to check if a document is annotated or not. The input MappedRecord `DocID` contains the document id as an instance of Long. The output values will be a Mapped Record `GenericFlag`.

The Input-Output Records for `IsAnnotated` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocID	MappedRecord	GenericFlag	MappedRecord	-

Description of the Input Record `DocID`:

Key	Value Type	Description	Default Value
doc_id*	Long	Document ID of the document that needs to be checked whether it is annotated or not.	-

Description of the Output Record `GenericFlag`:

Key	Value Type	Description	Default Value
flag	Boolean	Indicates that document is Annotated or not	-

The code sample to check, if a document is annotated or not, is:

```
try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("IsAnnotated");
//Create the input MappedRecord with name 'DocID'
```

```
MappedRecord input =
recordFactory.createMappedRecord("DocID");
//Specify the document id
Long docID = new Long("232554");
input.put("doc_id",docID);
//Execute the interaction
MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
//Retrieve the result flag
Boolean flag = (Boolean) output.get("flag");
System.out.println("IsAnnotated:" + flag.booleanValue());
}catch(ResourceException e){
      e.printStackTrace();
}
```

The error messages for `IsAnnotated` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid.	The input Record type is not a `DocID` MappedRecord.
FN_IS_RA_10354	Output Record type is invalid.	The output Record is not a `GenericFlag` MappedRecord.
FN_IS_RA_20389	The key 'doc_id' cannot be null in interaction IsAnnotated.	The key ‘doc_id’ is not specified in the input record.
FN_IS_RA_20390	The key 'doc_id' should be a Long in interaction IsAnnotated.	The ‘doc_id’ must be an instance of java.lang.Long.
FN_IS_RA_20391	The key 'doc_id' cannot be less than one (1) in interaction IsAnnotated	Invalid ‘doc_id’ is specified in the input record. This must be greater than or equal to 1.

GetAnnotations

GetAnnotations is used by the application component to retrieve annotation details of a document. The client will specify the document id, and page number. The input values are specified using the MappedRecord `DocumentPage`, and the output will be a Mapped Record of `Annotations`.

The Input-Output Records for `GetAnnotations` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocumentPage	MappedRecord	Annotations	MappedRecord	The output Record contains an instance of java.io.InputStream from

Input Record Description		Output Record Description		Remarks
				which the annotation information can be read for the specified docid and page number. The bytes read from the InputStream will be in XML format in accordance with XML Schema for Image Manager Annotations at Appendix B

Description of the Input Record `DocumentPage`:

Key	Value Type	Description	Default Value
doc_id*	Long	Document ID of document for which annotations have to be retrieved.	-
page_number*	Integer	Page Number of document for which annotations have to be retrieved. Specifying -1 returns annotations of all the pages.	-

Description of the Output Record `Annotations`:

Key	Value Type	Description	Default Value
XMLStream	InputStream	Annotations data in XML format as per XML Schema for Image Manager Annotations	-
ClientCodepage	String	Codepage used for Text Annotations data in XML	-

The code sample to retrieve the annotation data for the specified document id and page number is:

```
import java.io.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetAnnotations");
//Create the input MappedRecord with name 'DocumentPage'
MappedRecord input =
recordFactory.createMappedRecord("DocumentPage");
Long docID = new Long(232554);
Integer pageNumber = new Integer(1);
/* Specify Document id and page number whose annotations
need to be retrieved */
input.put("doc_id",docID);
input.put("page_number",pageNumber);
```

```
//Execute the interaction

MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);

//Retrieve the InputStream

InputStream xmlStream = (InputStream)
output.get("XMLStream");

//Retrieve the Codepage

String clientCodepage = (String)
output.get("ClientCodepage");

/* Read the stream data using any of the read methods of
java.io.InputStream class */

while(xmlStream.available()>0){

/* Read XML formatted annotation data that is compliant
with XML schema for Image Manager annotations */

}

/* Use the clientCodepage for decoding the annotation
data from byte array to String*/

xmlStream.close();

System.out.println("GetAnnotations completed
successfully");

}catch(ResourceException e){

      e.printStackTrace();

}catch(IOException e){

      e.printStackTrace();

}
```

The error messages for `GetAnnotations` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	Input Record must be `DocumentPage` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	Output Record must be Annotations MappedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type should be `DocumentPage`.
FN_IS_RA_20383	The key 'doc_id' cannot be null in interaction GetAnnotations	Doc id cannot be null in interaction GetAnnotations.
FN_IS_RA_20386	The key 'doc_id' should be a Long in interaction GetAnnotation	The specified doc_id object is not of type java.lang.Long
FN_IS_RA_20385	The key 'doc_id' cannot be less than one (1) in interaction GetAnnotations	The specified doc_id is either 0 or less than 0. Ensure that doc id has a value greater than 0.

Error Code	Error Message	Description
FN_IS_RA_20384	The key 'page_number' cannot be null in interaction GetAnnotations	Value for key ‘page_number’ is null. Ensure that the value is not null.
FN_IS_RA_20387	Value for key 'page_number' is invalid in interaction GetAnnotations.	The specified page number is either null or invalid value.
FN_IS_RA_20388	The key 'page_number' should be an Integer in interaction GetAnnotations	The specified page_number object is not of type java.lang.Integer
FN_IS_RA_11038	<90,0,6>Document not found by DOC	The DocID passed as the input parameter, either does not exist on the IS server or the user does not have permissions to view the document.
FN_IS_RA_11038	<80,0,18>Page number out of the range of pages in a document given to DOC	The requested page number does not exist. Enter a valid value for the page number you want to view. By default, each document has at least one page.
FN_IS_RA_11038	<90,2,5> Document that does not have view permissions for the logged in user.	The logged in user does not have permissions to view the page.

SaveAnnotations

SaveAnnotations interaction is used by the application component to save one or more annotation(s) in a document. This interaction is used to create new annotations and, update and delete existing annotations. The client specifies data for all annotations that needs to be created, updated or deleted, in the form of XML Stream. The input values are specified using the MappedRecord `Annotations`, and the output will be an Indexed Record `GenericResult`.

The Input-Output Records for `SaveAnnotations` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
Annotations	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `Annotations`:

Key	Value Type	Description	Default Value
XMLStream*	InputStream	Annotations data in XML format as per XML Schema for Image Manager Annotations.	-

Description of the Output Record `GenericResult`:

Key	Value Type	Description	Default Value
result	Long	Error code is returned, if returned from IS	0
ResultHashMap	HashMap	This HashMap will contain the necessary information as required by Daeja according to the documentation posted on the CSS web site http://www.css.filenet.com/products.asp?id=1112995173&fldName=ISRA+3%2E2&main	

Key	Value Type	Description	Default Value
		FolderID=1041356765&pathname=/Product%20Info/1)%20Products/Image%20Manager%20(IM)/Image%20Services%20Resource%20Adapter/Product%20Documentation/ISRA+3%2E2.If the annotation list is empty then the hashMap will also be empty.	

Description of the Output Record `ResultHashMap`:

Key	Value Type	Value	Description
new_annotations	String	"#new annotations" or null	In case one or more than one annotations are created, the value will be"#new annotations" else " null"
delete_annotations	String	"#delete annotations" or Null	In case one or more than one annotations are created and deleted, the value will be"#delete annotations" else "null"
modify_annotations	String	"#modify annotations" or Null	In case one or more than one annotations are created and modified, the value will be "#modify annotations" else "null"
annot_create_date	Date		In case one or more than one annotations are created, the value will be "System Date" else "null"
status	String	"OK" or "FAIL" or Null	In case one or more than one annotations are created and no error is generated while saving annotations, the value will be "OK" else " null"
response	String	"OK" or "NOT OK"	In case of success, the value will be "OK" else "NOT OK". The value for this parameter would be returned in case of any of the operations performed on annotations, i.e., Create/Modify/Delete.
annot_Id	Vector		In case one or more than one annotations are created, this Vector would contain list of elements else would be empty. Each element in the vector would be an object of HashMap. Each HashMap would contain two keys: • old_ID • new_ID The value corresponding to the old_ID will be the id generated by Daeja. The value corresponding to new_ID will be the Id generated by IS.

Description of each HashMap element contained in the Vector "annot_Id" retrieved from the "ResultHashMap" is:

Key	Value Type	Value	Description
old_ID	String		Value passed by Daeja for the new Annotation created
new_ID	String		Id generated by IS for the new annotation created

The code sample to save annotation data is:

```
import java.util.*;
import java.io.*;
try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("SaveAnnotations");
//Create the input MappedRecord with name 'Annotations'
MappedRecord input =
recordFactory.createMappedRecord("Annotations");
InputStream xmlStream = null;
/* Create the InputStream from which the annotation data
can be read by ISRA */
//Specify the created InputStream object
input.put("XMLStream",xmlStream);
//Execute the interaction
MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
//Retrieve the ResultHashMap
HashMap mappedAnnotIds =
(HashMap)output.get("ResultHashMap");
//Retrieve the values from mappedAnnotIds
System.out.println("value of new_annotations is
"+mappedAnnotIds.get("new_annotations"));
System.out.println("value of delete_annotations is "+
mappedAnnotIds.get("delete_annotations"));
System.out.println("value of modify_annotations is "+
mappedAnnotIds.get("modify_annotations"));
System.out.println("value of annot_create_date is
"+mappedAnnotIds.get("annot_create_date"));
System.out.println("value of status is
"+mappedAnnotIds.get("status"));
System.out.println("value of response is
"+mappedAnnotIds.get("response"));
//Retrieving the Vector from MappedAnnotIds
Vector AnnotIds= (Vector)
mappedAnnotIds.get("annot_Id"));
//Retrieving all the Vectors from AnnotIds
for(int i =0; i < AnnotIds.size(); i++)
 {
```

```
HashMap tempMappedAnnotIds = (HashMap)AnnotIds.get(i);
//Retrieving ids from tempMappedAnnotIds HashMap
System.out.println("value of old_ID is
"+tempMappedAnnotIds.get("old_ID"));
System.out.println("value of new_ID is
"+tempMappedAnnotIds.get("new_ID"));
}
// Retrieve the result error tuple
Long result = (Long)output.get("result");
long errorTuple = result.longValue();
if(errorTuple == 0)
System.out.println("Annotations saved successfully");
else
System.out.println("Error in saving annotations:" +
errorTuple);
}catch(ResourceException e){
      e.printStackTrace();
}
```

The error messages for `SaveAnnotations` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction	This interaction is not available in ISRA View Edition.
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not an `Annotations` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `GenericResults` MappedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	Name of input MappedRecord should be `'Annotations'`
FN_IS_RA_20381	The key 'XMLStream' is null	Specified `XMLStream` is null, ensure that it contains valid XML.
FN_IS_RA_20382	The key 'XMLStream' must be an instance of java.io.InputStream	The `'XMLStream'` must be an instance of java.io.InputStream.
FN_IS_RA_11039	<80,0,4>No permission to perform specified DOC function.	User does not have required permissions for saving annotations.
FN_IS_RA_11039	<90,0,6>Document not found by DOC.	The DocID passed, as the input parameter either does not exist on the IS server or the user does not have permissions to view the document.
FN_IS_RA_11039	<80,0,18>Page number out of the range of pages in a document given to DOC.	The requested page number does not exist. Enter a valid value for the page number you want to view. By default, each document has at least one page.

Error Code	Error Message	Description
FN_IS_RA_11039	<80,0,21>Annotation not found by DOC	The annotation to be modified/deleted does not exist. Ensure that annotation to be updated or deleted exists on document.
FN_IS_RA_11039	<80,0,23>DOC annotation is busy being updated by another client.	Annotation is already locked by another client/user.
FN_IS_RA_11039	<80,0,26>Annotation not busy when override was requested of DOC	The request for annotation lock has been made with override lock as true, on an annotation that was not locked by any other client/user.
FN_IS_RA_11054	Annotation Data exceeds the maximum limit of 800 Bytes	Length of annotation data is exceeding the limit of 800 bytes.

GetDocumentContent2

GetDocumentContent2 interaction is used by the application component to retrieve the content of multiple pages of the requested Document.

The user has to specify the Document-ID and the first page number to be retrieved. The last page number is a non-mandatory field. If the client specifies last page number parameter, the content of the pages retrieved is for the specified range. If the last page number is null or out of range, the content of pages starting from the first page number specified to the last page number, is retrieved.

The GetDocument interaction results in the migration of the physical pages from the optical disk to the default cache or the specified cache.

The pre-fetch count can be specified to pre-fetch the number of additional pages from the optical disk.

The polling interval is the frequency (in seconds) at which ISRA checks to determine if the document has been migrated to cache or not. The valid value for polling interval is between 250-5000 milliseconds.The polling interval can also be specified in milliseconds for this interaction. The valid value for polling interval (in ms) is in between 250-5000 milliseconds. The default value is 5000 milliseconds.

The Record `DocContentRequest`, of the type `MappedRecord`, is used to specify the input values. The content of the first page is returned as a stream of bytes while the content of rest of the pages is retrieved in a HashMap as pagenumber-bytearray (key/value) pair contained in the Record `DocContent`, which is of type `MappedRecord`.

The Input-Output Records for `GetDocumentContent` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocContentRequest	MappedRecord	DocContent	MappedRecord	The output record contains an object of type InputStream and HashMap, which contains content of rest of the pages as pagenumber-bytearray pair. This class provides methods to read the stream of bytes (as binary data).

Description of the Input Record `DocContentRequest`:

Key	Value Type	Remarks
doc_id*	Long	The Document-ID. It is a mandatory field.
page_number*	Integer	The page number of the document. It is a mandatory field.
last_page_number	Integer	The last page number of the document, which the client wishes to retrieve. It can be less than the total number of pages in the document. It is a non mandatory field
page_cache	String	Name of the IS page cache to migrate the page to. If not specified, the default cache is used. It is an optional value. Page cache name should be in <object name>: <domain name>: <organization name> format.
prefetch_count	Integer	Number of additional pages to pre-fetch from the optical disk to increase the relative access speed of the document. It is an optional parameter. Default value is 0.
polling_interval	Integer	The polling interval used to check if the document has been migrated from the optical storage device. Default value is 5.
polling_interval_ms	Integer	The polling interval used to check if the document has been migrated from the optical storage device. Valid value is in between 250-5000 ms. Default value is 5000.
no_of_times_to_poll	Integer	Number of times to poll. Default value is 1.
checksum_enabled	Boolean	Flag to determine if Checksumming to be done for the document being retrieved. Default value is false.

Description of Output Record `DocContent`:

Key	Value Type	Remarks
mime_type	String	MIME/content type of the returned content.
file_name	String	File name associated with the document.
buffer_pages	HashMap	It contains the content of all the pages as pagenumber – bytearray pair.

Note: For documents that are NOT present in the cache, the application component might get an 'Object does not exist error, <77,0,10>'. In such a scenario, the application component would re-issue the GetDocumentContent request, after waiting for some duration. In ISRA3.0a, a new parameter, *no_of_times_to_poll*, has been added. ISRA polls for n number of times (where n is *no_of_times_to_poll*) where each polling time is t secs (where t is *polling_interval*).

The points to consider when the GetDocument interaction is executed are:

- While requesting the retrieval of a particular page of a document, the Document-ID , first page number and the last page number of the document must be specified through the keys 'doc_id', 'page_number' and 'last_page_number' This should be done using the input MappedRecord `DocContentRequest`.
- A specific `page_cache` can be specified to migrate the pages by using the 'page_cache' key of the `DocContentRequest`. By specifying the cache name, access to the

page would be faster. This is because the page whose content is to be read may already reside in this page_cache. Thus, the user will not have to wait for the page to migrate from the optical disk to the cache.

- If both polling_interval and polling_interval_ms are specified then polling_interval will take precedence over polling_ interval_ms.

Note: If the page cache is not specified, the document will be migrated to the default cache of the domain to which the user is logged in.

- The number of documents to be pre-fetched from the optical disk can be specified, using the key 'prefetch_count'. This assumes that the user wants to access the pages that follow the page number specified in the key 'page_number'. Executing this interaction will give a total number (1 + prefetch_count) of pages to be brought into the cache. This speeds up the access to the subsequent pages.

The code sample to get document content is:

```
import java.util.*;


try {
interactionSpec.setFunctionName("GetDocumentContent2");
MappedRecord inputRecord =
recordFactory.createMappedRecord ("DocContentRequest");
//Insert values into the input record.
inputRecord.put("doc_id", new Long(200123));
inputRecord.put("page_number", new Integer(1));
inputRecord.put("last_page_number", new Integer(4));
inputRecord.put("page_cache","page_cache1:FNIS:FileNet");
inputRecord.put("prefetch_count", new Integer(4));
inputRecord.put("polling_interval", new Integer(4));
inputRecord.put("polling_interval_ms", new
Integer(4000));
inputRecord.put("no_of_times_to_poll", new Integer(2));
inputRecord.put("checksum_enabled", new Boolean(true));
/*Invoke the interaction with execute method that returns
the output record of type mapped record.* /
MappedRecord outputRecord = (MappedRecord)
interaction.execute(interactionSpec, inputRecord);
String fileName = (String) outputRecord.get("file_name");
String mimeType = (String) outputRecord.get("mime_type");
Map restOfPages = new HashMap();
```

```
restOfPages = (Map) outputRecord.get(“buffer_pages”);
int firstpageNo = 1;
int lastpageNo = 4;
for(int i=firstpageNo;i<=lastpageNo;i++)
{
      byte[] bydata = null;
      bydata= (byte[])restOfPages.get(new
      Integer(i).toString());
}
System.out.println("Mime type of the returned content:" +
mimeType);
System.out.println("Name of the file:" + fileName);
}catch (ResourceException e) {
       e.printStackTrace();
}
```

The error messages for `GetDocumentContent` Interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10360	The key 'doc_id' cannot be null in interaction GetDocumentContent.	Doc id cannot be null in interaction GetDocumentContent.
FN_IS_RA_10361	Domain name and/or Organization name in the key 'page_cache' is not valid.	The page cache name as specified should belong to the same domain where you are logged on. No cross-domain cache names are allowed. For example, if you are logged on to ‘DomainA:OrganizationA’, then a valid cache name could be ‘page_cache1: DomainA: OrganizationA’, not ‘page_cache1:DomainB:OrganizationB’.
FN_IS_RA_10362	TimeOut cannot be less than polling_interval.	The TimeOut value cannot be less than polling interval.
FN_IS_RA_10363	<80,0,2> Document not found by Doc.	The DocID passed, as the input parameter either does not exist on the IS server or the user does not have permissions to view the document.
FN_IS_RA_10363	Invalid cache name specified.	The specified cache does not exist. Check the correct name from your IS administrator. For example, page_cache1:DomainA:OrganizationA, where DomainA: OrgnaizationA, refers to the domain to which you are logged on.
FN_IS_RA_10383	The key 'prefetch_count' cannot be less than zero (0) in interaction GetDocumentContent.	Specified value should be greater than zero.
FN_IS_RA_10384	The key 'polling_interval'	Specify a valid polling interval value in the range (1,

Error Code	Error Message	Description
	should be greater than 250 milliseconds and less than equal to five (5) seconds.	5) (inclusive of 1 and 5).
FN_IS_RA_10385	The key 'doc_id' cannot be less than one (1) in interaction GetDocumentContent.	Specified value should be greater than zero.
FN_IS_RA_10389	Doc Id object should be object of type Long in interaction GetDocumentContent.	DocId should be object of type Long in interaction GetDocumentContent.
FN_IS_RA_10391	Polling Interval should be an Integer in interaction GetDocumentContent.	Polling Interval should be an Integer.
FN_IS_RA_10392	Prefetch Count should be an Integer in interaction GetDocumentContent.	Prefetch Count should be an Integer.
FN_IS_RA_11071	IS calculated Checksum value and Local Checksum value is not identical. Data might be corrupt!	The checksum value stored in the IS for the particular document does not match the checksum value calculated by ISRA. This error signifies that there is a high probability of the document data being corrupt.
FN_IS_RA_40071	The key 'first_page_number' should be an Integer in interaction GetDocumentContent	First Page Number should be an Integer.
FN_IS_RA_40072	The key 'first_page_number' cannot be null in interaction GetDocumentContent	First Page Number cannot be null.
FN_IS_RA_40073	The key 'first_page_number' cannot be less than one (1) in interaction GetDocumentContent.	The first page number cannot be zero or an empty value.
FN_IS_RA_40074	The key 'last_page_number' should be an Integer in interaction GetDocumentContent	Last Page Number should be an Integer.
FN_IS_RA_40076	The key 'last_page_number' cannot be less than one (1) in interaction GetDocumentContent.	The last page number cannot be a negative value
FN_IS_RA_40077	The key 'last_page_number' cannot be less than first_page_number in interaction GetDocumentContent	The last page number cannot be less that the first page number
FN_IS_RA_40078	The key 'first_page_number' is out of Range in interaction GetDocumentContent	The first page number is greater than the total number of pages in the document.
FN_IS_RA_40080	Retrieved the Byte Array of length 0, End of Document	There are no more pages in the document to be retrieved , the byte array of length 0 signifies the end

Error Code	Error Message	Description
	Reached.	of document.

Folder Interactions

GetFolderFolders

The GetFolderFolders interaction is used by the application component to retrieve sub folders and their properties from a specified folder. The input record `FolderRequest` is a MappedRecord, containing the name of the folder and max_limit, which is the maximum number of sub folders to be returned. The output record `FolderFolders` is a MappedRecord, which contains a `ResultSet`, containing the properties of the sub folders present in the folder and the base folder name.

The Input-Output Records for `GetFolderFolders` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
FolderRequest	MappedRecord	FolderFolders	MappedRecord	-

Description of the Input Record `FolderRequest`:

Key	Value Type	Description	Default Value
folder_name*	FolderName	Name of the folder the contents of which would be retrieved.	-
max_limit*	Long	Maximum number of items to return.	-

Description of the Output Record `FolderFolders`:

Key	Value Type	Description
base_folder_name	FolderName	Name of the folder the contents of which the user requested.
contained_folders	ResultSet	Forward scrollable set of FolderProperties Records.

Note: For 'max_limit' value zero, the `ResultSet` in the output record contains one row. However, the `ResultSet` does not contain any row, if the specified folder name is invalid or, if the user does not have necessary security permissions on the folder.

The code sample to retrieve the properties of the sub folders inside a specified folder is:

```
import java.sql.SQLException;

try {
//Define the type of interaction.
interactionSpec.setFunctionName ("GetFolderFolders");
```

```
//Create the input mapped record.

MappedRecord inputRecord=
recordFactory.createMappedRecord(“FolderRequest");

//Insert values into the input record.

inputRecord.put ("folder_name", "valid_folder_name");

inputRecord.put ("max_limit", new Integer(100));

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord contains a
ResultSet, and the base folder name. */

MappedRecord outputRec = null;

outputRec =
(MappedRecord)interaction.execute(interactionSpec,
inputRecord);

// Now retrieve the ResultSet object

ResultSet resultSet =
(ResultSet)outputRec.get(“contained_folderesultSet”);

while (resultSet.next())

{

/*All the defined propertied for folder can be accessed
from resultset rsresultSet. */

long folderId = resultSet.getLong("folder_id");

System.out.println("FodlerId is:" + folderId);


String folderName = (String)
resultSet.getString("folder_name");

System.out.println("FolderName is:" + folderName);

}

}catch (ResourceException e) {

      e.printStackTrace();

}catch(SQLException e){

      e.printStackTrace();

}
```

The error messages for `GetFolderFolders` interaction are listed in the following table:

Error Code	Error Message	Description

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_10393	<90,0,53>Invalid folder name	The correct syntax for the folder name is "/Account". Make sure that the folder name contains a '/' in the beginning.
FN_IS_RA_10393	The folder length exceeds 152 characters.	The folder name can contain a maximum of 152 characters, including the preceeding '/' character.
FN_IS_RA_10393	A sub-folder length exceeds 18 characters.	A sub-folder length is limited to a maximum of 18 characters including the slash.
FN_IS_RA_10393	<90,0,49> Attempt to create too many folder levels.	There can be a maximum of 8 folder levels, each separated by a slash.
FN_IS_RA_10414	Invalid folder name.	Specified Folder name is either null or not a `String` or blank String.
FN_IS_RA_10422	Max Limit should be an Integer in interaction GetFolderFolders.	Max Limit object should be an Integer object in GetFolderFolders interaction.

GetFolderAttributes

This interaction will enable the client to retrieve attribute values of specified folder. The client will specify the folder id and/or folder name of the folder whose properties are to be returned. The input values would be specified through the MappedRecord, FolderAttributeRequest, and the output would be a MappedRecord.

The execute () method in the Interaction object would call the getFolderAttributes () method on ISInterface layer. The ISInterface layer would call the getFolderAtts() method of the Service Layer class, FN_IS_RPC_INX_Service, to complete the Interaction.

The Input-Output Records for `GetFolderAttributes` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
FolderAttributeRequest	MappedRecord	FolderProperties	MappedRecord	

Description of the Input Record `FolderAttributeRequest`:

Key	Value Type	Description	Default Value
folder_id	Long	Id of the folder	
folder_name	String	Name of the folder	

Description of the Output Record `FolderProperties`:

Key	Value Type	Description
folder_name	String	Name of the folder
folder_id	Long	Folder id
is_leaf	Boolean	Is leaf folder

Key	Value Type	Description
is_closed	Boolean	Status of the folder.
sec_read	Long	Security read
sec_write	Long	Security write
sec_append_execute	Long	Security append execute
date_create	Date	Date of creation
date_archive	Date	Archival date
date_delete	Date	Date of deletion
auto_del_period	Short	auto_del_period
retent_base	Short	retent_base
retent_offset	Long	retent_offset
retent_disposition	Short	retent_disposition
is_nonleaf	Boolean	Is non-leaf folder

The code sample to retrieve attribute values of specified folder is:

```
import java.util.*;


try {

//Define the type of interaction.

interactionSpec.setFunctionName (“GetFolderAttributes”);

//Create the input mapped record.

MappedRecord inputRecord =
recordFactory.createMappedRecord
(“FolderAttributeRequest”);

//Insert values into the input record.

inputRecord.put ("folder_name", "valid_folder_name");

inputRecord.put ("folder_id",new Long(12345));

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord which contains the
folder attributes. */

MappedRecord outputRec = null;

outputRec =
(MappedRecord)interaction.execute(interactionSpec,
inputRecord);

/* Extract the elements of the MappedRecord if and only
if the interaction is a success. */

Map folderProperties = new HashMap();
```

```
folderProperties.putAll(outputRec);

System.out.println("FolderName is:" +
folderProperties.get("folder_name"));

System.out.println("FolderID is:" +
folderProperties.get("folder_id"));

System.out.println("FolderName is:" +
folderProperties.get("is_leaf"));

System.out.println("is nonleaf :" +
folderProperties.get("is_nonleaf"));

System.out.println("is_closed:" +
folderProperties.get("is_closed"));

System.out.println("sec_read:" +
folderProperties.get("sec_read"));

System.out.println("sec_write:" +
folderProperties.get("sec_write"));

System.out.println("sec_append_execute:" +
folderProperties.get("sec_append_execute"));

System.out.println("Date_create:" +
folderProperties.get("date_create"));

System.out.println("Date_archive:" +
folderProperties.get("date_archive"));

System.out.println("Date_delete:" +
folderProperties.get("date_delete"));

System.out.println("Auto_del_period is:" +
folderProperties.get("auto_del_period"));

System.out.println("Retent_base is:" +
folderProperties.get("retent_base"));

System.out.println("Retent_offset is:" +
folderProperties.get("retent_offset"));

System.out.println("Retent_disposition is:" +
folderProperties.get("retent_disposition"));

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for `GetFolderAttributes` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10424	Invalid folder id	Folder Id specified is invalid
FN_IS_RA_10426	Invalid folder id and name specified.	Folder Id and Folder Name specified are not concurrent.
FN_IS_RA_10414	Invalid folder name.	Folder Name specified is invalid.
FN_IS_RA_10425	Error occurred in interaction GetFolderAttributes.	Common error tuple for an error in the GetFolderAttributes interaction.

CreateFolders

This interaction creates a new folder either at root level or in one of the existing folders. The client will specify the folder name and folder properties. This interaction returns the folder ID of a newly created folder. The input values would be specified through the MappedRecord, FolderProperties, and the output would be a MappedRecord.

The execute() method in the Interaction object would call the createFolders() method on ISInterface layer. The ISInterface layer would call the createFolder() method of the Service Layer class, FN_IS_RPC_INX_Service, to complete the Interaction.

The Input-Output Records for `CreateFolders` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
FolderProperties	MappedRecord	GenericResult	MappedRecord	

Description of the Input Record `FolderProperties :`

Key	Value Type	Description	Default Value
folder_name*	String	User-assigned name for the folder.	
AccessRestrictions	String	Security permissions on folder. It is the id corresponding to the user name. To obtain the id for a user, use the following command: Sec_tool>nametoid <username> To obtain the name for a user id, use the following command: Sec_tool>decode <userID>	1,1,1
auto_del_period	Short	From date filed, the number of months until a document is eligible for automatic unfiling from the folder.	0
retent_base	Short	Close Date or Entry Date (creation date): Specifies when to start counting the retention period. The default is Entry Date.. 49 – For Entry Date 0 – For close Date.	49

Key	Value Type	Description	Default Value
retent_offset	Long	Specifies the length of the retention period in months from the base date, i.e., either the Entry Date or Close Date.	996 months is the default value.
retent_disposition	Short	Archive or Delete: At the end of the retention period, a folder can be archived or deleted. The default is Delete. 49 – For Archive 0 – For Delete	0

Description of the Output Record `GenericResult`:

Key	Value Type	Description
Result	Long	Incase of success this will be a system-assigned identification number for the folder created. In case of error, it will be the FileNet error tuple.

The code sample to retrieve attribute values of specified folder is:

```
import java.util.*;


try{

//Define the type of interaction.

interactionSpec.setFunctionName (“CreateFolder”);

//Create the input mapped record.

MappedRecord inputRecord =
recordFactory.createMappedRecord (“FolderProperties”);

//Insert values into the input record.

inputRecord.put ("folder_name", "test");

inputRecord.put("AccessRestrictions" ,"1,1,1");

inputRecord.put("retent_base", new Short((short)49));

inputRecord.put("retent_offset", new Long(23));

inputRecord.put("retent_disposition", new
Short((short)0));

inputRecord.put("auto_del_period", new Integer(10));

/*Invoke the interaction with execute method that returns
the output record of type MappedRecord which contains the
folder id. */

MappedRecord outputRec = null;
```

```
outputRec =
(MappedRecord)interaction.execute(interactionSpec,
inputRecord);

/* Extract the elements of the MappedRecord if and only
if the interaction is a success. */

Map folderProperties = new HashMap();

folderProperties.putAll(outputRec);

System.out.println("FolderID is:" +
folderProperties.get("result"));

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for CreateFolder interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_40065	Folder properties cannot be null.	Folder Id specified is invalid
FN_IS_RA_10414	Invalid folder name.	Folder Name specified is invalid.
FN_IS_RA_40046	Invalid value for key 'AccessRestrictions'.	The key 'AccessRestrictions' must be an object of type String.
FN_IS_RA_40050	Auto Delete Period must be a Short Object.	The key “auto_del_period” must be an object of type short.
FN_IS_RA_40051	Folder Retent Base must be a Short Object.	Common error tuple for an error in the CreateFolder interaction.
FN_IS_RA_40052	Folder Retent Offset must be a Long Object.	The key "retent_offset" must be an object of type Long.
FN_IS_RA_40053	Folder Retent Disposition must be a Short Object.	The key "retent_disposition" must be an object of type Short.

DeleteFolders

This interaction will enable the client to delete existing folders. The client will specify the folder name to be deleted along with the information to un-file the documents from the folder. The input values would be specified through the IndexedRecord, DeleteFolderSet, and the output would be an IndexedRecord containing the error tuple for each folder deleted.

The execute() method in the Interaction object would call the deleteFolders() method on ISInterface layer. The ISInterface layer would call the deleteFolder() method of the Service Layer class, FN_IS_RPC_INX_Service, to complete the Interaction.

The Input-Output Records for `DeleteFolderSet` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
DeleteFolderSet	IndexedRecord	FolderErrorSet	IndexedRecord	

Description of the Input Record `DeleteFolderSet :`

Key	Value Type	Description	Default Value
folder_name*	String	User-assigned name for the folder.	
unfile_docs*	Boolean	If documents are to be deleted.	

Description of the Output Record `FolderErrorSet:`

Key	Value Type	Description
folder_name	String	User-assigned name for the folder.
error_code	Long	Error code

The code sample to retrieve attribute values of specified folder is:

```
import java.util.*;


try {
// Define the type of interaction.
interactionSpec.setFunctionName (“DeleteFolder”);
//Create the input Indexed record.
IndexedRecord inputRecord =
recordFactory.createIndexedRecord (“DeleteFolderSet”);
//Create a MapperRecord to send folder information
MappedRecord eachRecord =recordFactory.createMappedRecord
(“DeleteFolderEachRecord”);
//Insert each folder information into MappedRecord
eachRecord.put ("folder_name", "valid_folder_name");
eachRecord.put ("unfile_docs",new Boolean(“true”));
//Insert the MappedRecord into input record
inputRecord.add(eachRecord);
/*Invoke the interaction with execute method that returns
the output record of type IndexedRecord which contains
the folder attributes. */
IndexedRecord outputRec = null;
```

```
outputRec =
(IndexedRecord)interaction.execute(interactionSpec,
inputRecord);

/* Extract the elements of the IndexedRecord if and only
if the interaction is a success. */

Iterator iterator = outputRec.iterator();

while(iterator.hasNext()){

  Map map1 = new HashMap();

  map1.putAll((Map)iterator.next());

  System.out.println("FolderName is:" +
map1.get("folder_name"));

     System.out.println("Error Code is:" +
map1.get("error_code"));

}

}catch(ResourceException e){

      e.printStackTrace();

}
```

The error messages for `DeleteFolders` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_40065	Folder properties cannot be null.	Folder Id specified is invalid
FN_IS_RA_40055	Delete Folder Set is empty.	Folder Id and Folder Name specified not concurrent.
FN_IS_RA_40054	Delete Folder Description is Null.	Folder Name specified is invalid.
FN_IS_RA_40068	Delete Folders record entry is not an instance of MappedRecord.	Common error tuple for an error in the GetFolderAttributes interaction.
FN_IS_RA_40056	Error in deleting folders.	Generic error code, for any exception which occurs while executing the delete folder interaction.
FN_IS_RA_40057	The key 'folder_name' is Null.	Specified Folder name is null.
FN_IS_RA_40058	The key 'folder_name' must be a String object.	Specified Folder name is not a `String` object.
FN_IS_RA_40059	The key 'folder_name' is empty.	Specified Folder name is a blank String.
FN_IS_RA_40060	The key 'unfile_docs' is Null.	The key 'unfile_docs' cannot be Null.
FN_IS_RA_40061	The key 'unfile_docs' must be a Boolean object.	The key 'unfile_docs' must be an object of type Boolean.

UpdateFolders

This interaction will enable the client to update the properties of an existing folder. The client will specify the folder name of the folder whose properties are to be updated. The input values would be specified through the MappedRecord, UpdateFolder, and the output would be a MappedRecord containing the GenericResult.

The execute () method in the Interaction object would call the updateFolder() method on ISInterface layer. The ISInterface layer would call the updateFolder () method of the Service Layer class, FN_IS_RPC_INX_Service, to complete the Interaction.

The Input-Output Records for `UpdateFolder` are explained in the following table:

Input Record Description		**Output Record Description**		**Remarks**
Name	**Type**	**Name**	**Type**	**-**
UpdateFolder	MappedRecord	GenericResult	MappedRecord	

Description of the Input Record `UpdateFolder:`

Key	**Value Type**	**Description**	**Default Value**
folder_name*	String	User-assigned name for the folder.	
folder_id*	Long	System-assigned identification number for this folder.	
is_leaf	Boolean	Is leaf folder	true
is_closed	Boolean	Is folder closed	
AccessRestrictions	String	Security permissions on folder It is the id corresponding to the user name. To obtain the id for a user, use the following command: Sec_tool>nametoid <username> To obtain the name for a user id, use the following command: Sec_tool>decode <userID>	1,1,1
date_create	Date	Date on which the folder was created	
date_archive	Date	Date on which the folder becomes eligible for archiving	
date_delete	Date	Date on which the folder can be deleted	
auto_del_period	Short	From date filed, the number of months until a document is eligible for automatic unfiling from the folder.	0
retent_base	Short	Close Date or Entry Date (creation date): Specifies when to start counting the retention period. The default is Entry Date.. 49 – For Entry Date 0 – For close Date.	49

Key	Value Type	Description	Default Value
retent_offset	Long	Specifies the length of the retention period in months from the base date, i.e., either the Entry Date or Close Date.	996 months is the default value.
retent_disposition	Short	Archive or Delete: At the end of the retention period, a folder can be archived or deleted. The default is Delete. 49 – For Archive 0 – For Delete	0

Description of the Output Record `FolderProperties`:

Key	Value Type	Description
Result	Long	This will be the FileNet error tuple.

The code sample to retrieve attribute values of specified folder is:

```
try {
// Define the type of interaction.
interactionSpec.setFunctionName (“UpdateFolder”);
//Create the input mapped record.
MappedRecord mappedRecord =
recordFactory.createMappedRecord (“UpdateFolder”);
//Insert values into the input record.
inputRecord.put ("folder_name", "valid_folder_name");
inputRecord.put( "folder_id", new Long(50351));
inputRecord.put( "is_leaf", new Boolean(true));
inputRecord.put( "is_closed", new Boolean(false));
inputRecord.put( "AccessRestrictions", new
String("2,2,2"));
inputRecord.put( "date_create", null);
inputRecord.put( "date_archive", null);
inputRecord.put( "date_delete", null);
inputRecord.put( "auto_del_period", new Short("12"));
inputRecord.put( "retent_base", new short(49));
inputRecord.put( "retent_offset", new Long(12));
```

```
inputRecord.put( "retent_disposition", new short(0));
/*Invoke the interaction with execute method that returns
the output record of type MappedRecord which contains the
folder attributes. */

MappedRecord outputRec = null;

outputRec =
(MappedRecord)interaction.execute(interactionSpec,
inputRecord);

/* Extract the elements of the MappedRecord if and only
if the interaction is a success. */

System.out.println("Error Code is:" +
outputRec.get("result "));

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for `UpdateFolder` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10414	Invalid folder name.	Folder Name specified is invalid.
FN_IS_RA_10424	Invalid folder id	Folder Id specified is invalid
FN_IS_RA_40043	Invalid value for key 'is_leaf'.	The key 'is_leaf' must be an object of type Boolean.
FN_IS_RA_40045	The key 'is_closed' must be a Boolean object.	The key 'is_closed' must be an object of type Boolean.
FN_IS_RA_40046	Invalid value for key 'AccessRestrictions'.	The key 'AccessRestrictions' must be an object of type String.
FN_IS_RA_40047	Invalid value for key 'date_create'.	The key 'date_create'must be an object of type Date.
FN_IS_RA_40048	Invalid value for key 'date_archive'.	The key 'date_archive'must be an object of type Date.
FN_IS_RA_40049	Invalid value for key 'date_delete'.	The key 'date_delete' must be an object of type Date.
FN_IS_RA_40050	Auto Delete Period must be a Short Object.	The key “auto_del_period” must be an object of type Short.
FN_IS_RA_40051	Folder Retent Base must be a Short Object.	The key ‘retent_base’ must be an object of type Short.
FN_IS_RA_40052	Folder Retent Offset must be a Long Object.	The key ‘retent_offset’ must be an object of type Long.
FN_IS_RA_40053	Folder Retent Disposition must be a Short Object.	The key ‘retent_disposition’ must be an object of type Short.
FN_IS_RA_40081	Auto Delete Period must be greater than or equal to zero.	Auto Delete Period must be a positive integer value.

Error Code	Error Message	Description
FN_IS_RA_40082	Folder Retent Offset must be greater than or equal to zero.	Retent offset must be a positive integer value.
FN_IS_RA_40083	Error occurred in interaction UpdateFolder.	Generic error code, for any exception which occurs while executing the update folder interaction.

Queue Interactions

GetWorkspaces

The GetWorkspaces interaction is used by the application component to query for the list of workspaces in the configured IS. The client will send an empty anonymous Record instance, i.e., an instance of Record without any specific name and data. The output record will be an IndexedRecord called `Workspaces`, which will contain list of workspace names as configured in the IS.

The Input-Output Records for `GetWorkspaces` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
-	MappedRecord	Workspaces	IndexedRecord	The input record is null or an unnamed (anonymous) Record object. The output IndexedRecord will contain workspace names as String objects.

Description of the Output Record `Workspaces`:

The output IndexedRecord would contain workspace names.

The code sample to retrieve all the workspaces is:

```
import java.util.*;

try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetWorkspaces");
//Create the unnamed input MappedRecord
MappedRecord input =
recordFactory.createMappedRecord(“”);
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* Iterate through the returned IndexedRecord and display
the workspace names */
```

```
ListIterator iterator = output.listIterator();
while(iterator.hasNext()){
      String workspaceName = (String)iterator.next();
      System.out.println(workspaceName);
}
}catch (ResourceException e) {
       e.printStackTrace();
}
```

The error messages for `GetWorkspaces` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.

GetQueues

The GetQueues interaction is used by the application component to query for the list of queues in a specified workspace or all the queues in the configured IS.

The input would be a `MappedRecord`, `WorkspaceName`, containing the key `'workspace_name'`. If the client specifies the workspace name, then the queues in that workspace are returned. If, workspace name is not specified, then the list of all WorkFlo workspaces and queues in the IS would be returned.

The output record will be an IndexedRecord called `Queues`, which will contain a Map object for the specified workspace. If no workspace was specified in the input record, then the output record will contain Map objects for all the workspaces available in the IS. Each Map object contains a list of queue names.

The Input-Output Records for `GetQueues` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
WorkspaceName	MappedRecord	Queues	IndexedRecord	If no workspace name is specified in the input record, the output IndexedRecord will hold Map objects corresponding to each workspace in the IS. If a workspace name is specified, then the output IndexedRecord contains one Map object for the specified workspace.

Description of the Input Record `WorkspaceName`:

Key	Value Type	Description	Default Value
workspace_name	`String`	The workspace name whose queues are to be retrieved.	

Description of the Output Record Queues:

This is an IndexedRecord that will contain a Map instance for the specified workspace. If no workspace was specified, it will contain one Map instance for each workspace.

Each Map will contain one key-value pair, the key is a String object which is name of the workspace. The value is a List that contains queue names as String objects.

The code sample to retrieve queue names for the specified workspace is:

```
import java.util.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetQueues");
//Create the input MappedRecord with name “WorkspaceName”
MappedRecord input =
recordFactory.createMappedRecord("WorkspaceName");
//Specify the workspace name
input.put("workspace_name","Workspace1");
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
//Extract the Map object for the specified workspace
Map workspaceMap = (Map) output.get(0);
//Retrieve the workspace name, which is the key in the
Map
Iterator iterator = workspaceMap.keySet().iterator();
while(iterator.hasNext()){
      String workspaceName = (String)iterator.next();
      System.out.println(workspaceName);
      List queueList =
(List)workspaceMap.get(workspaceName);
Iterator queueNamesItr = queueList.iterator();
While(queueNamesItr.hasNext())
      System.out.println(queueNamesItr.next());
}
}catch (ResourceException e) {
       e.printStackTrace();
}
```

The error messages for GetQueues interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10396	No valid license to execute the interaction.	This interaction is available only in ISRA Enterprise Edition.
FN_IS_RA_20352	<152,2,16>The name's object does not exist.	The Workspace name specified does not exist on the IS. Check for the correct Workspace name.
FN_IS_RA_20352	<156,2,13>Syntax error in an object field.	The Workspace name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace name.

GetQueueFields

The GetQueueFields interaction is used by application components to retrieve the field descriptions of a queue. The client will specify the required workspace and queue names. The input values are specified using the MappedRecord, QueueName, and the output will be an IndexedRecord, QueueFieldDescription. The output record will contain several Map objects, corresponding to each user field, in the queue.

The Input-Output Records for GetQueueFields are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
QueueName	MappedRecord	QueueFieldDescription	IndexedRecord	The output record would contain Map objects corresponding to each user field in the queue.

Description of the Input Record QueueName:

Key	Value Type	Description	Default Value
workspace_name*	String	The workspace name.	-
queue_name*	String	The queue name	-

Description of the Map objects contained in Output Record: QueueFieldDescription:

Key	Value Type	Description	Default Value
field_name	String	The field name.	-
field_type	Integer	The data type of the field.	-
field_length	Integer	The maximum length of the field data.	-
key_type	Integer	The value indicates whether the field contains unique values or not.	-
is_required	Boolean	The value indicates whether the field required or not.	-
is_rendezvous	Boolean	The value indicates whether the field is a rendezvous queue.	
can_be_displayed	Boolean	The value indicates if the field can be displayed or not.	

The code sample to retrieve queue field information is:

```
import java.util.*;


try{
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetQueueFields");
//Create the input MappedRecord with name “QueueName”
MappedRecord input =
recordFactory.createMappedRecord("QueueName");
//Specify the workspace name and queue name
input.put("workspace_name","Workspace1");
input.put("queue_name","Queue1");
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* Iterate through the list of field descriptions and
display the details */
Iterator listIterator = output.listIterator();
While(listIterator.hasNext()){
Map fieldDescriptionMap = (Map)listIterator.next();
System.out.println(fieldDescriptionMap.get("field_name"))
;
System.out.println(fieldDescriptionMap.get("field_type"))
;
System.out.println(fieldDescriptionMap.get("field_length"
));
System.out.println(fieldDescriptionMap.get("key_type"));
System.out.println(fieldDescriptionMap.get("is_required")
);
System.out.println(fieldDescriptionMap.get("is_rendevous"
));
System.out.println(fieldDescriptionMap.get("can_be_displa
yed"));
}
}catch (ResourceException e) {
      e.printStackTrace();
}
```

Tip: Refer to Appendix A: References for Queue Field Type Definitions.

The error messages for `GetQueueFields` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_20362	The key 'workspace_name' is null.	Specified Workspace name is null.
FN_IS_RA_20364	The key 'queue_name' is null.	Specified Queue name is null.
FN_IS_RA_20363	The key 'workspace_name' must be a String object.	Specified Workspace name is not a String object (Workspace name object should be of type java.lang.String).
FN_IS_RA_20365	The key 'queue_name' must be a String object.	Specified Queue name is not a String object (Queue name object should be of type java.lang.String).
FN_IS_RA_20354	<151,0,35>The specified workspace does not exist.	The Workspace name specified does not exist on the IS. Check for the correct Workspace name.
FN_IS_RA_20354	<151,0,7>Undefined queue.	The Queue name specified does not exist on the IS. Check for the correct Queue name.
FN_IS_RA_20354	<151,0,22>Security violation.	The user does not have access to the specified Queue.
FN_IS_RA_20354	<156,2,13>Syntax error in an object field.	The Workspace or Queue name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace and Queue names.

GetQueueEntries

The `GetQueueEntries` interaction is used by the application component to retrieve one or more entries in a queue, which satisfy some input criteria. The client specifies a SQL style query, the maximum number of entries to be returned, and whether to mark the rows returned as busy or not. The input values are specified using the MappedRecord, `QueueRequest`, and the output will be a `ResultSet`.

The Input-Output Records for `GetQueueEntries` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
QueueRequest	MappedRecord	QueueQueryResult	ResultSet	-

Description of the Input Record `QueueRequest`:

Key	Value Type	Description	Default Value
query*	String	The query string. Query string would hold queue name, WHERE and ORDER BY clause. The minimal query will be "SELECT * FROM [QueueName]"	-

Key	Value Type	Description	Default Value
max_rows*	Integer	The maximum number of rows to be returned.	-
set_busy*	Boolean	If set, the returned rows would be marked as busy.	-
delete_spec*	Integer	deleteNone – 0, deleteAfterRead – 1	-

Description of Output Record `QueueQueryResult:`

The output would be a forward scrollable set of queue entries.

Description of input query format:

SELECT * FROM [WorkspaceName/QueueName]

WHERE [user_field_name_1] = [user_field_value_1]

AND [user_field_name_2] = [user_field_value_2]

AND [user_field_name_N] = [user_field_value_N]

AND F_TIMEOUT < [user-specified-date]

AND F_PRIORITY > [user-specified-min-priority]

AND F_PRIORITY <= [user-specified-max-priority]

AND F_GROUPID= [user-specified-group-name]

AND F_USERID = [user-specified-user-name]

AND F_BUSY = {TRUE | FALSE}

ORDER BY [user_field_name] {ASC | DESC}

Note: Fields of type String, Date or Time must be specified within single or double quotes. The minimal query is “SELECT * FROM [WorkSpaceName/QueueName] “

Refer to Appendix A: References for system fields that can be used in the query.

The code sample to retrieve queue entries for the specified queue is:

```
import java.sql.SQLException;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetQueueEntries");
//Create the input MappedRecord with name “QueueRequest”
MappedRecord input =
recordFactory.createMappedRecord("QueueRequest");
```

```
//Specify the query and other required parameters
input.put("query","select * from Workspace1/Queue1");
input.put("max_rows", new Integer(10));
input.put("set_busy", new Boolean(false));
input.put("delete_spec", new Integer(0));
//Execute the interaction
ResultSet resultSet = (ResultSet)
interaction.execute(interactionSpec,input);
/* Traverse through the ResultSet and display the queue
entry details */
while(resultSet.next()){
          int userId = resultSet.getInt("F_USERID");
     System.out.println(userId);
}
}catch (ResourceException e) {
      e.printStackTrace();
}catch(SQLException e){
      e.printStackTrace();
}
```

The error messages for `GetQueueEntries` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_20367	The key 'query' is null.	Query cannot be null as it is a mandatory parameter in this interaction.
FN_IS_RA_20369	The key 'max_rows' is null.	The max rows parameter should be greater than or equal to one (1), this is a mandatory value.
FN_IS_RA_20370	The key 'set_busy' is null.	set_busy cannot be null as it is a mandatory parameter in this interaction.
FN_IS_RA_20372	The key 'delete_spec' is null.	Delete_spec cannot be null as it is a mandatory parameter in this interaction.
FN_IS_RA_20368	The key 'query' must be a String object.	Specified query is not a String object (query object should be of type java.lang.String).
FN_IS_RA_20371	The key 'set_busy' must be a Boolean object.	Specified set_busy is not a Boolean object (set_busy should be of type java.lang.Boolean).
FN_IS_RA_10368	The key 'max_rows' cannot be less than one (1).	The max rows parameter should be greater than or equal to one (1).
FN_IS_RA_20373	The key 'delete_spec' must be an Integer object.	Specified delete_spec is not an Integer object (delete_spec should be of type java.lang.Integer).
FN_IS_RA_11052	Invalid queue query.	The query specified is not syntactically correct.
FN_IS_RA_11042	Invalid ORDER BY clause.	Specified ORDER BY clause in the query is invalid.

Error Code	Error Message	Description
FN_IS_RA_10002	Invalid column name.	A field name, which is not present in the queue definition, is used.
FN_IS_RA_11043	Only F_PRIORITY can occur twice in the query.	A field name, other than F_PRIORITY is used more than once in the query. Only F_PRIORITY is allowed more than once in the query syntax.
FN_IS_RA_11044	Column specified twice in the query.	A field name, other than F_PRIORITY is used more than once in the query. Only F_PRIORITY is allowed more than once in the query syntax.
FN_IS_RA_11045	Invalid F_USERID match condition specified in the query.	The operator used with F_USERID field is invalid.
FN_IS_RA_11046	Invalid F_PRIORITY specified in the query.	Invalid value used for F_PRIORITY (Allowed range of values for F_PRIORITY is 0 to 9)
FN_IS_RA_11047	Invalid F_PRIORITY match condition specified in the query.	The operator used with F_PRIORITY field is invalid.
FN_IS_RA_11059	Invalid F_BUSY match condition specified in the query.	The operator used with F_BUSY field is invalid.
FN_IS_RA_11060	Invalid F_TIMEOUT match condition specified in the query.	The operator used with F_TIMEOUT field is invalid.
FN_IS_RA_11058	Only equals operator('=') can be specified for user defined fields in the query.	Only equals operator can be used with user fields in the query.
FN_IS_RA_11061	Invalid value for Number field specified in the query.	Invalid value for Number field specified in the query.
FN_IS_RA_11062	Invalid value for Document field specified in the query.	Invalid value for Document field specified in the query.
FN_IS_RA_11063	Invalid value for Folder field specified in the query.	Invalid value for Folder field specified in the query.
FN_IS_RA_11065	Invalid value for Decimal field specified in the query.	Invalid value for Decimal field specified in the query.
FN_IS_RA_11066	Invalid value for Boolean field specified in the query.	Invalid value for Boolean field specified in the query.
FN_IS_RA_11064	Invalid value for Integer field specified in the query.	Invalid value for Integer field specified in the query.
FN_IS_RA_20353	<151,0,35> The specified workspace does not exist.	The Workspace name specified does not exist on the IS. Check if the Workspace name is correct.
FN_IS_RA_20353	<151,0,7> Undefined queue.	The Queue name specified does not exist on the IS. Check if the Queue name is correct.
FN_IS_RA_20353	<151,0,22> Security violation.	The user does not have access to the specified Queue.
FN_IS_RA_20353	<156,2,13>Syntax error in an object field.	The Workspace or Queue name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace and Queue names.

InsertQueueEntries

The InsertQueueEntries interaction is used by the application component to insert one or more entries in a queue. The client will specify the queue name and the entries to be inserted in the queue. The input values are specified using the MappedRecord, `QueueInsertRequest`, and the output will be the MappedRecord, called `GenericResult`.

Description of the Input-Output Record for `InsertQueueEntries`:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
QueueInsertRequest	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `QueueInsertRequest`:

Key	Value Type	Description	Default Value
workspace_name*	String	Workspace name.	-
queue_name*	String	Queue name	-
queue_entries*	IndexedRecord	List of entries to be inserted in the queue. Each entry is an IndexedRecord (QueueEntry) of QueueFields (MappedRecord).	-

QueueEntry

A QueueEntry is an IndexedRecord of QueueFields.

Description of the `QueueField` MappedRecord:

Key	Value Type	Description	Default Value
field_name*	String	Name of the field.	-
field_value	Object	The exact type of value depends upon the field type in queue definition, which can be retrieved using GetQueueFields interaction.	-

Description of Output Record, `GenericResult`:

Key	Value Type	Description	Default Value
result	Long	This will be the FileNet error tuple.	-

The code sample to insert queue entries in the specified queue is:

```
try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("InsertQueueEntries");
/* Create the input MappedRecord with name
“QueueInsertRequest” */
```

```
MappedRecord input =
recordFactory.createMappedRecord("QueueInsertRequest");

String workspaceName = "Workspace1";

String queueName = "Queue1";

//Create the “QueueEntry” record

IndexedRecord queueEntries =
recordFactory.createIndexedRecord("QueueEntry");

/* Create “QueueField” records and specify the field
names and field values */

MappedRecord queueField1 =
recordFactory.createMappedRecord("QueueField");

queueField1.put("field_name","userField1");

queueField1.put("field_value",new Integer(23));

MappedRecord queueField2 =
recordFactory.createMappedRecord("QueueField");

QueueField2.put("field_name","userField2");

QueueField2.put("field_value",new Integer(23));

//Add the queue field details into an IndexedRecord

IndexedRecord queueEntry =
recordFactory.createIndexedRecord("QueueEntry");

queueEntry.add(queueField1);

queueEntry.add(queueField2);

queueEntries.add(queueEntry);

/* Specify the workspace name, queue name and the queue
entry object in the input record */

input.put("workspace_name",workspaceName);

input.put("queue_name",queueName);

input.put("queue_entries",queueEntries);

//Execute the interaction

MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);

//Retrieve the result error tuple

Long result = (Long)output.get("result");

long errorTuple = result.longValue();

if(errorTuple == 0)

     System.out.println("Queue entries inserted
successfully");

else
```

```
        System.out.println("Error in inserting Queue
entries:" + errorTuple);
}catch (ResourceException e) {
            e.printStackTrace();
}
```

The error messages for `InsertQueueEntries` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_20362	The key 'workspace_name' is null.	Workspace name cannot be null as it is a mandatory parameter in InsertQueueEntries interaction.
FN_IS_RA_20364	The key 'queue_name' is null.	Queue name cannot be null as it is a mandatory parameter in InsertQueueEntries interaction.
FN_IS_RA_20363	The key 'workspace_name' must be a String object.	Specified Workspace name is not a String object (Workspace name object should be of type java.lang.String).
FN_IS_RA_20365	The key 'queue_name' must be a String object.	Specified Queue name is not a String object (Queue name object should be of type java.lang.String).
FN_IS_RA_20375	The key 'queue_entries' is null.	queue_entries cannot be null as it is a mandatory parameter in InsertQueueEntries interaction.
FN_IS_RA_20376	The key 'queue_entries' must be an instance of javax.resource.cci.IndexedRecord.	The specified queue_entries is not an IndexedRecord. (queue_entries object should be of type javax.resource.cci.IndexedRecord.)
FN_IS_RA_20356	<151,0,35>The specified workspace does not exist.	The specified Workspace name does not exist on the IS. Check if the Workspace name is correct.
FN_IS_RA_20356	<151,0,7>Undefined queue.	The specified Queue name does not exist on the IS. Check if the Queue name is correct.
FN_IS_RA_20356	<151,0,8>Undefined user field.	The user field specified does not exist in specified queue on the IS. Check if the user field is correct.
FN_IS_RA_20356	<151,0,11>Field specified more than once.	The user field specified more than once in the input parameter.
FN_IS_RA_20356	<151,0,41>An entry already exists with the same value for a unique field.	An entry already exists with the same value for a unique field.
FN_IS_RA_20356	<151,0,10>Invalid system field data - priority value must be between WQS_min_priority and WQS_max_priority.	Invalid F_PRIORITY value specified in input parameter. Allowed range of values for F_PRIORITY is 0 to 9.
FN_IS_RA_20356	<151,0,9>Invalid system field.	The specified system field is not valid.
FN_IS_RA_20356	<151,0,22>Security violation.	The user does not have access to the specified Queue.
FN_IS_RA_20356	<156,2,13>Syntax error in an object field.	The Workspace or Queue name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace and Queue names.

Error Code	Error Message	Description
FN_IS_RA_11068	Specified field value is incompatible with field type.	Specified field value is incompatible with field type.

DeleteQueueEntries

DeleteQueueEntries is used by the application component to delete one or more entries from a queue. The client will specify the queue name and entries to be deleted from this queue. The input values are specified through the MappedRecord, `QueueEntryIDSet`, and the output will be the MappedRecord, called `GenericResult`.

The Input-Output Records for `DeleteQueueEntries` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
QueueEntryIDSet	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `QueueEntryIDSet`:

Key	Value Type	Description	Default Value
workspace_name*	String	Workspace name.	-
queue_name*	String	Queue name.	-
queue_entries*	IndexedRecord	List of entries in the queue to be deleted. Each entry Id is a String that corresponds to an entry in a queue.	-

Description of the Output Record `GenericResult`:

Key	Value Type	Description	Default Value
result	Long	This will be the FileNet error tuple.	-

The code sample to delete queue entries from the specified queue is:

```
import java.util.*;
import java.sql.SQLException;


try {
/* Execute GetQueueEntries interaction to find the Queue
entry IDs */
List queueEntriesList = new ArrayList();
interactionSpec.setFunctionName("GetQueueEntries");
MappedRecord input =
recordFactory.createMappedRecord("QueueRequest");
//Find all the queue entries whose F_PRIORITY=9
```

```
input.put("query","select * from Workspace1/Queue1 where
F_PRIORITY=9");

input.put("max_rows", new Integer(10));

input.put("set_busy", new Boolean(false));

input.put("delete_spec", new Integer(0));

ResultSet resultSet = (ResultSet)
interaction.execute(interactionSpec,input);

while(resultSet.next()){

     String queueEntryID =
resultSet.getString("ENTRYID");

//Add the Queue Entry Ids in the list

     queueEntriesList.add(queueEntryID);

}

String workspaceName = "Workspace1";

String queueName = "Queue1";

interactionSpec = new FN_IS_CciInteractionSpec();

//Specify the function name for the interaction
interactionSpec.setFunctionName("DeleteQueueEntries");

/* Create the input MappedRecord with name
“QueueEntryIDSet” */

input =
recordFactory.createMappedRecord("QueueEntryIDSet");

IndexedRecord queueEntries =
recordFactory.createIndexedRecord("");

queueEntries.addAll(queueEntriesList);

input.put("workspace_name",workspaceName);

input.put("queue_name",queueName);

input.put("queue_entries",queueEntries);

//Execute the interaction

MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);

Long result = (Long)output.get("result");

long errorTuple = result.longValue();

if(errorTuple == 0)

     System.out.println("Queue entries deleted
successfully");

else
```

```
      System.out.println("Error in deleting Queue
entries:" + errorTuple);
}catch (ResourceException e) {
      e.printStackTrace();
}catch(SQLException e){
      e.printStackTrace();
}
```

Note: You need to execute the GetQueueEntries interaction, to find the Queue Entry Ids, before executing DeleteQueueEntries.

The error messages for `DeleteQueueEntries` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_20362	The key 'workspace_name' is null.	Workspace name cannot be null as it is a mandatory parameter in DeleteQueueEntries interaction.
FN_IS_RA_20364	The key 'queue_name' is null.	Queue name cannot be null as it is a mandatory parameter in DeleteQueueEntries interaction.
FN_IS_RA_20363	The key 'workspace_name' must be a String object.	Specified Workspace name is not a String object (Workspace name object should be of type java.lang.String).
FN_IS_RA_20365	The key 'queue_name' must be a String object.	Specified Queue name is not a String object (Queue name object should be of type java.lang.String).
FN_IS_RA_20375	The key 'queue_entries' is null.	queue_entries cannot be null as it is a mandatory parameter in InsertQueueEntries interaction.
FN_IS_RA_20376	The key 'queue_entries' must be an instance of javax.resource.cci.Indexed Record	Specified queue_entries is not a IndexedRecord (queue_entries object should be of type javax.resource.cci.IndexedRecord.)
FN_IS_RA_20355	<151,0,35>The specified workspace does not exist.	The specified Workspace name does not exist on the IS.
FN_IS_RA_20355	<151,0,7>Undefined queue.	The specified Queue name does not exist on the IS.
FN_IS_RA_20355	<151,0,26>Invalid entry_id.	The specified entry_id does not exist in the queue.
FN_IS_RA_20355	<151,0,22>Security violation.	The user does not have access to the specified Queue.
FN_IS_RA_20355	<156,2,13>Syntax error in an object field.	The Workspace or Queue name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace and Queue names.

UpdateQueueEntries

The UpdateQueueEntries interaction is used by the application component to update one or more entries in a queue. The client will specify the queue name and the entries to be updated

in the queue. The input values are specified using the MappedRecord, `QueueIDEntry`, and the output will be the MappedRecord, called `GenericResult`.

The Input-Output Records for `UpdateQueueEntries` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
QueueIDEntry	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `QueueIDEntry`:

Key	Value Type	Description	Default Value
workspace_name*	String	Workspace name.	-
queue_name*	String	Queue name.	-
queue_entry_id*	String	Entry Id.	-
queue_field*	IndexedRecord	QueueEntry.	-

QueueEntry

A QueueEntry is an IndexedRecord of QueueField records.

Description of `QueueField`:

Key	Value Type	Description	Default Value
field_name*	String	Field name.	-
field_value*	Object	Value depends upon the field type in queue definition.	-

Description of Output Record `GenericResult`:

Key	Value Type	Description	Default Value
result	Long	This will be the FileNet error tuple.	-

The code sample to update queue entries in the specified queue is.

```
import java.sql.SQLException;


try {
/* Execute GetQueueEntries interaction to find the Queue
entry IDs */
interactionSpec.setFunctionName("GetQueueEntries");
MappedRecord input =
recordFactory.createMappedRecord("QueueRequest");
//Find all the queue entries whose F_PRIORITY=9
```

```
input.put("query","select * from Workspace1/Queue1 where
F_PRIORITY=9");
input.put("max_rows", new Integer(10));
input.put("set_busy", new Boolean(false));
input.put("delete_spec", new Integer(0));
ResultSet resultSet = (ResultSet)
interaction.execute(interactionSpec,input);
String workspaceName = "Workspace1";
String queueName = "Queue1";
interactionSpec = new FN_IS_CciInteractionSpec();
//Specify the function name for the interaction
interactionSpec.setFunctionName("UpdateQueueEntries");
while(resultSet.next()){
     String queueEntryID =
resultSet.getString("ENTRYID");
     IndexedRecord queueEntries =
recordFactory.createIndexedRecord("QueueEntry");
/* Create the input MappedRecord with name
“QueueEntryIDSet” */
     input =
recordFactory.createMappedRecord("QueueIDEntry");
     MappedRecord queueField1 =
recordFactory.createMappedRecord("QueueField");
     queueField1.put("field_name","userField1");
     queueField1.put("field_value",new Integer(23));
     queueEntries.add(queueField1);
     input.put("workspace_name",workspaceName);
     input.put("queue_name",queueName);
     input.put("queue_entry_id",queueEntryID);
     input.put("queue_field",queueEntries);
//Execute the interaction
     MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
     Long result = (Long)output.get("result");
     long errorTuple = result.longValue();
     if(errorTuple == 0)
          System.out.println("Queue entries updated
successfully");
```

```
        else
                System.out.println("Error in updating Queue
entries:" + errorTuple);
         }
}catch (ResourceException e) {
        e.printStackTrace();
}catch(SQLException e){
        e.printStackTrace();
}
```

The error messages for `UpdateQueueEntries` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_20362	The key 'workspace_name' is null.	Workspace name cannot be null as it is a mandatory parameter in UpdateQueueEntries interaction.
FN_IS_RA_20364	The key 'queue_name' is null.	Queue name cannot be null as it is a mandatory parameter in UpdateQueueEntries interaction.
FN_IS_RA_20363	The key 'workspace_name' must be a String object.	Specified Workspace name is not a String object (Workspace name object should be of type java.lang.String).
FN_IS_RA_20365	The key 'queue_name' must be a String object.	Specified Queue name is not a String object (Queue name object should be of type java.lang.String).
FN_IS_RA_20377	The key 'queue_entry_id' is null.	Queue_entry_id cannot be null as it is a mandatory parameter in UpdateQueueEntries interaction.
FN_IS_RA_20378	The key 'queue_entry_id' must be a String object.	Specified queue_entry_id is not a String object (queue_entry_id object should be of type java.lang.String).
FN_IS_RA_20379	The key 'queue_field' is null.	queue_field cannot be null as it is a mandatory parameter in UpdateQueueEntries interaction.
FN_IS_RA_20380	The key 'queue_field' must be an instance of javax.resource.cci.IndexedRecord.	Specified queue_field is not a IndexedRecord (queue_field object should be of type javax.resource.cci.IndexedRecord.)
FN_IS_RA_20357	<151,0,35>The specified workspace does not exist.	The specified Workspace name does not exist on the IS.
FN_IS_RA_20357	<151,0,7>Undefined queue.	The specified Queue name does not exist on the IS.
FN_IS_RA_20357	<151,0,26>Invalid entry_id	The specified entry_id does not exist in the queue.
FN_IS_RA_20357	<151,0,22>Security violation.	The user does not have access to the specified Queue.
FN_IS_RA_20357	<151,0,41>An entry already exists with the same value for a unique field.	An entry already exists with the same value for a unique field

Error Code	Error Message	Description
FN_IS_RA_20357	<151,0,10>Invalid system field data - priority value must be between WQS_min_priority and WQS_max_priority.	Invalid F_PRIORITY value specified in input parameter. Allowed range of values for F_PRIORITY is 0 to 9.
FN_IS_RA_20357	<156,2,13>Syntax error in an object field.	The Workspace or Queue name specified contains a colon (:). Specify only one part name. Do not append domain and organization names to the Workspace and Queue names.
FN_IS_RA_11068	Specified field value is incompatible with field type.	Specified field value is incompatible with field type.
FN_IS_RA_11067	User field specified in the query is invalid.	The specified user field is invalid.

CreateWorkspace

This interaction will allow users to create a workspace on IS. User will provide the name of the workspace, description of the workspace & access permissions for the workspace.

The execute () method in the Interaction object would call the createWorkSpace () method on ISInterface layer. The ISInterface layer would call the createWorkSpace () method of the Service Layer class, FN_IS_RPC_WQS_Service, to complete the Interaction.

The Input-Output Records for `CreateWorkspace` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	**-**
WorkSpaceDescription	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `createWorkspaceRequest`:

Key	Value Type	Description	Default Value
workspace_name *	String	Name of the workspace	
Access_restrictions	String	Read/write/AppendExecute permissions for the workspace, for e, “SysAdmin,SysAdmin,SysAdmin”.	“ANYONE,ANYONE, ANYONE”
text_desc	String	Description of the workspace.	NULL

Description of Output Record `GenericResult`:

Key	Value Type	Description
result	Long	This will be the FileNet error tuple.

The code sample to create a workspace on IS:

```
try{
/* Execute CreateWorkspace interaction to create a
workspace on IS */
interactionSpec.setFunctionName("CreateWorkSpace");
```

```
MappedRecord input =
recordFactory.createMappedRecord("WorkSpaceDescription
");
// Add parameter (Map) to be passed to IS RA */
input.put("workspace_name","workspace1");
input.put("Access_restrictions",
“SysAdmin,SysAdmin,SysAdmin”);
input.put("text_desc",”create workspace1”);
//Execute the interaction
     MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
     Long result = (Long)output.get("result");
     long errorTuple = result.longValue();
     if(errorTuple == 0)
          System.out.println("Workspace created
successfully");
     else
          System.out.println("Error while creating
Workspace:" + errorTuple);
 }
}catch (ResourceException e) {
        e.printStackTrace();
}
```

The error messages for `CreateWorkspace` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_40005	Error occurred while creating workspace.	Common error tuple for error in CreateWorkspace interaction.
FN_IS_RA_40007	The key 'security_permissions' must be a String object.	Specified 'security_permissions' is not a String object ('security_permissions' object should be of type java.lang.String).
FN_IS_RA_40008	The key 'text_desc' must be a String object.	Specified ‘text description’ is not a String object (‘text description’ object should be of type java.lang.String).
FN_IS_RA_40028	The key 'workspace_name' is empty.	Workspace name cannot be null as it is a mandatory parameter in createWorkspace interaction.
FN_IS_RA_40032	Workspace name length exceeds 14 characters.	The length of the workspace name specified should either be equal to 14 characters or less than 14 characters.
FN_IS_RA_40039	Workspace description exceeds 800 characters.	The length of the workspace description specified should either be equal to 800 characters or less than 800 characters.

CreateQueue

This interaction will allow users to create a queue on IS. User will provide the name of the workspace, name of the queue, description access permissions & content access permissions along with the list of the user fields associated with this queue.

The execute () method in the Interaction object would call the createQueue () method on ISInterface layer. The ISInterface layer would call the createQueue() method of the Service Layer class, FN_IS_RPC_WQS_Service, to complete the Interaction.

The Input-Output Records for `CreateQueue` are explained in the following table:

Input Record Description		**Output Record Description**		**Remarks**
Name	**Type**	**Name**	**Type**	**-**
QueueDescription	MappedRecord	GenericResult	MappedRecord	-

Description of the Input Record `CreateQueueRequest`:

Key	**Value Type**	**Description**	**Default Value**
workspace_name*	String	Name of the workspace	
queue_name*	String	Name of the queue.	
content_access	String	Read/write/AppendExecute permissions for content of queue for example, “SysAdmin,SysAdmin,SysAdmin”.	“ANYONE,ANYONE, ANYONE”
definition_access	String	Read/write/AppendExecute permissions for description of queue	“ANYONE,ANYONE, ANYONE”
text_desc	String	Description of the queue.	NULL
user_field_desc *	IndexedRecord	List of user fields to be inserted. Each entry in the IndexedRecord (UserFieldDesc) is UserField MappedRecord	

A user field contains various attributes for each field.

Key	**Value Type**	**Description**	**Default Value**
fieldName *	String	Field name.	
fieldType *	Short	Field type in queue definition.	
unique	Short	Possible values can be 0, 1, 2	0
fieldLength *	Short	Length of field for different field types.	
isRequired	Boolean	True, if it is a mandatory field.	false
isRendezvous	Boolean	True, if it is a rendezvous field.	false
canBeDisplayed	Boolean	True, if it is to be displayed.	true

Description of Output Record `GenericResult`:

Key	Value Type	Description
result	Long	This will be the FileNet error tuple.

Valid Queue field types:

Field Type	Value
Number	1
String	2
Time	3
Selection	4
Document	5
Folder	6
Int	7
Date	8
Boolean	10
Null	11
Decimal	12

The code sample to create a queue on IS:

```
import java.util.*;


try{
/* Execute CreateQueue interaction to create a workspace
on IS */
interactionSpec.setFunctionName("CreateQueue");
MappedRecord input =
recordFactory.createMappedRecord("QueueDescription");
// Creating indexed record for all the Queue Fields.
IndexedRecord queueFieldDescList=
recordFactoryrecordFactory.createIndexedRecord("QueueFiel
dDescList");
// Creating mapped record for each Queue Field
description.
MappedRecord eachQueueFieldDesc =recordFactory
recordFactory.createMappedRecord("EachQueueFieldDesc");
Map userField = new HashMap();
userField.put("fieldName", "field1");
userField.put("fieldType", new Short("2"));
userField.put("fieldLength",new Short("40"));
```

```
userField.put("unique",new Short("0"));
userField.put("isRequired",new Boolean(false));
userField.put("isRendevous",new Boolean(false));
userField.put("canBeDisplayed",new Boolean(true));
eachQueueFieldDesc.putAll(userField);

//Add Each queue field description to the list
queueFieldDescList.add(eachQueueFieldDesc);
// Add parameter (Map) to be passed to IS RA */
input.put("workspace_name", “Workspace1”);
input.put("queue_name", “Queue1”);
input.put("definition_access",
“SysAdmin,SysAdmin,SysAdmin”);
input.put("content_access",
“SysAdmin,SysAdmin,SysAdmin”);
input.put("text_desc", “Create Queue”);
input.put("user_field_desc", queueFieldDescList);
//Execute the interaction
     MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
     Long result = (Long)output.get("result");
     long errorTuple = result.longValue();
     if(errorTuple == 0)
          System.out.println("Queue created
successfully");
     else
          System.out.println("Error while creating
Queue:" + errorTuple);
 }
}catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `CreateQueue` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_40009	Error occurred while creating queue.	Common error tuple for error in CreateQueue Interaction.
FN_IS_RA_40011	The key 'definition_access'	Specified 'definition_access' is not a String object

Error Code	Error Message	Description
	must be a String object.	('definition_access' object should be of type java.lang.String).
FN_IS_RA_40012	The key 'content_access' must be a String object.	Specified 'content_access' is not a String object ('content _access' object should be of type java.lang.String).
FN_IS_RA_40013	The key 'user_field_desc' is null.	'user_field_desc' cannot be null as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40014	The key 'user_field_desc' must be an IndexedRecord object.	Specified 'user_field_desc' is not a String object ('user_field_desc' object should be of type java.lang.String).
FN_IS_RA_40015	The key 'fieldName' is null.	'fieldName' cannot be null as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40016	The key 'fieldName' must be a String object.	Specified 'fieldName' is not a String object ('fieldName' object should be of type java.lang.String).
FN_IS_RA_40017	The key 'fieldType' is null.	'fieldType' cannot be null as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40018	The key 'fieldType' must be a Short object.	Specified 'fieldType' is not a Short object ('fieldType' object should be of type java.lang.Short).
FN_IS_RA_40019	The key 'fieldLength' is null.	'fieldLength' cannot be null as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40020	The key 'fieldLength' must be a Short object.	Specified 'fieldLength' is not a Short object ('fieldLength' object should be of type java.lang.Short).
FN_IS_RA_40021	The key 'unique' must be a Short object.	Specified 'unique' is not a Short object ('unique' object should be of type java.lang.Short).
FN_IS_RA_40022	The key 'isRequired' must be a Boolean object.	Specified 'isRequired' is not a Boolean object ('isRequired' object should be of type java.lang.Boolean).
FN_IS_RA_40023	The key 'isRendevous' must be a Boolean object.	Specified 'isRendevous' is not a Boolean object ('isRendevous' object should be of type java.lang.Boolean).
FN_IS_RA_40024	The key 'canBeDisplayed' must be a Boolean object.	Specified 'canBeDisplayed' is not a Boolean object ('canBeDisplayed' object should be of type java.lang.Boolean).
FN_IS_RA_40025	The key 'fieldName' is empty.	'fieldName' cannot be empty as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40026	The key 'fieldType' is empty.	'fieldType' cannot be empty as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40027	The key 'fieldLength' is empty.	'fieldLength' cannot be empty as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40028	The key 'workspace_name' is empty.	‘workspace_name' cannot be empty as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40029	The key 'queue_name' is empty.	'queue_name' cannot be empty as it is a mandatory parameter in createQueue interaction.
FN_IS_RA_40030	The key 'user_field_desc' is empty.	'user_field_desc' cannot be empty as it is a mandatory parameter in createQueue interaction.

Error Code	Error Message	Description
FN_IS_RA_40032	Workspace name length exceeds 14 characters.	The length of the workspace name specified should either be equal to 14 characters or less than 14 characters.
FN_IS_RA_40033	Queue name length exceeds 14 characters.	The length of the Queue name specified should either be equal to 14 characters or less than 14 characters.
FN_IS_RA_40037	Duplicate Queue Field Name specified.	Duplicate queue field name cannot be specified in the createQueue interaction.
FN_IS_RA_40038	Invalid Queue Field Type specified.	
FN_IS_RA_40040	Queue description exceeds 800 characters.	The length of the Queue description specified should either be equal to 800 characters or less than 800 characters.
FN_IS_RA_40041	Queue field name length exceeds 18 characters.	The length of the Queue field name specified should either be equal to 18 characters or less than 18 characters.

Meta Data Interactions

GetDocClassIndices

The GetDocClassIndices interaction is used by the application component to retrieve all indices associated with a document class, including their associated data types.

The name of the document class is specified using the MappedRecord `DocClassName`. The output is returned as an IndexedRecord `DocClassIndex`, which contains Map instances.

The Input-Output Records for `GetDocClassIndices` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocClassName	MappedRecord	DocClassIndex	IndexedRecord containing instances of java.util.Map.	The output IndexedRecord contains Map instances corresponding to each index of the document class.

Description of Input Record `DocClassName`:

Key	Value Type	Remarks
docclass_name*	String	Name of the Document Class.

Description of the Map instances contained in output Record `DocClassIndex`:

Key	Value Type	Remarks
index_name	String	Name of the Index field of the class specified in the input record.
index_type	Short	The data type of the index field. See Appendix A for valid values and

Key	Value Type	Remarks
		corresponding Java data types.
required	Boolean	Indicates if this index is mandatory.

The code sample to get the Document Class Indices is:

```
import java.util.*;


try {
interactionSpec.setFunctionName("GetDocClassIndices");
MappedRecord inputRecord =
recordFactory.createMappedRecord ("DocClassName");
inputRecord.put("docclass_name","Account");
// Returned indexed record contains instances of Map.
IndexedRecord outputRecord =(IndexedRecord)
interaction.execute(interactionSpec, inputRecord);
ListIterator iterator = outputRecord.listIterator();
/* Iterate through the indexed record to retrieve each
map instance. */
while(iterator.hasNext())
{
Map map = (Map) iterator.next();
//Retrieve the values of index_type and index_name.
String indexName = (String)map.get("index_name");
Short indexType = (Short)map.get("index_type");
Boolean isRequired = (Boolean)map.get("required");
System.out.println ("Index name:" + indexName + " Index
Type: " + indexType+ " Is required: " + isRequired);
}
}catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `GetDocClassIndices` interaction are listed in the following table:

Error Code	Exception Message	Description
FN_IS_RA_10398	Specified document class does not exist.	The Document Class name does not exist.

Error Code	Exception Message	Description
FN_IS_RA_10401	Invalid document class name.	The document class is either null, or not an object of type `java.lang.String` or blank string.

GetMenuValue

GetMenuValue is used by the application component to retrieve menu value for the specified menu label.

Specify the name of the index field, on which the menu applies along with the menu label, using the `MenuValueRequest` MappedRecord. The ISRA returns the menu value associated with the menu label in a `MenuValueReturn` MappedRecord.

The Input-Output Records for `GetMenuValue` are explained in the following table:

Input Record Description		Output Record Description	
Name	**Type**	**Name**	**Type**
MenuValueRequest	MappedRecord	MenuValueReturn	MappedRecord

Description of Input Record `MenuValueRequest`:

Key	Value Type	Remarks
index_name*	String	Name of the index field on which the menu label applies.
menu_label*	String	The menu label for which the value is requested. Used in both input and output records.

Description of Output Record `MenuValueReturn`:

Key	Value Type	Remarks
menu_label	String	The menu label for which the client is requesting the value.
menu_value	Integer	The menu value, associated with the menu label given as input.

The code sample to get Menu Value is:

```
try  {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetMenuValue");
MappedRecord inputRecord =
recordFactory.createMappedRecord ("MenuValueRequest");
inputRecord.put("index_name", "AutoClaimList");
inputRecord.put("menu_label", "AC3");
//Execute the interaction
MappedRecord outputRecord = (MappedRecord)
interaction.execute(interactionSpec, inputRecord);
String menuLabel = (String)
outputRecord.get("menu_label");
```

```
Integer menuValue = (Integer)
outputRecord.get("menu_value");

System.out.println ("Menu Label:" + menuLabel + " Menu
Value:" + menuValue);

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for the `GetMenuValue` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10365	Specified index of type menu does not exist.	The index name either does not exist on IS or is not of type Menu.
FN_IS_RA_10371	The key 'index_name' or 'menu_label' cannot be null.	Either the Index or the menu label whose value is to be retrieved is null. Ensure that both are not null or empty.
FN_IS_RA_10394	Invalid index name.	The index name is either null, or not an object of type `java.lang.String` or blank string.
FN_IS_RA_10395	Invalid menu label.	The menu label is either null, or not an object of type `java.lang.String` or blank string.

GetSecurityInfo

GetSecurityInfo is used by the application component to get the information from the IS security database about system, group, user and devices. The input MappedRecord, `SecurityInputRecord`, will contain values corresponding to the keys, object_id and/or object_name, or none of them. The output will be an IndexedRecord, `SecurityInfo`.

If the client does not supply input values for either of the keys in the input MappedRecord, then the output record will contain data for all the users and groups in the IS, otherwise it would have data of the user/group whose name or ID is passed as input.

The Input-Output Records for `GetSecurityInfo` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
SecurityInputRecord	MappedRecord	SecurityInfo	IndexedRecord	The output IndexedRecord would hold MappedRecord objects corresponding to each object name or object id.

Description of Input Record `SecurityInputRecord`:

Key	Value Type	Description	Default Value
object_name	String	The object name (three part name).	-

Key	Value Type	Description	Default Value
object_id	Integer	The object id.	-

Description of MappedRecord instances contained in Output Record `SecurityInfo`:

Key	Value Type	Description	Default Value
object_name	String	The object name.	-
object_id	Long	The object id.	-
primary_group	Long	Primary object group.	-
members_in	Vector	List of group id's of which the user is a member .	-

The code sample to retrieve security info for the specified object is:

```
import java.util.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetSecurityInfo");
/* Create the input MappedRecord with name
“SecurityInputRecord” */
MappedRecord input =
recordFactory.createMappedRecord("SecurityInputRecord");
//Specify the three part object name
input.put("object_name","user1:fnbuild:FileNet");
input.put("object_id", new Integer(23));


//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* If no input parameters were specified, then the output
IndexedRecord will be a Collection of detailed Map
objects for all the users and group. Otherwise the
IndexedRecord will contain only one Map object */
ListIterator iterator = output.listIterator();
//Extract the info out of the IndexedRecord
while(iterator.hasNext()){
      Map securityDetailsMap = (Map) iterator.next();
      System.out.println("==============");
      System.out.println("Object name:" +
securityDetailsMap.get("object_name"));
```

```
        System.out.println("Object id:" +
securityDetailsMap.get("object_id"));
        System.out.println("Primary Group:" +
securityDetailsMap.get("primary_group"));
        Vector groupIDs = (Vector)
securityDetailsMap.get("members_in");
        System.out.println("Group IDs");
        Iterator groupIDIterator = groupIDs.iterator();
        while(groupIDIterator.hasNext())
              System.out.println(groupIDIterator.next());
}
}catch (ResourceException e) {
        e.printStackTrace();
}
```

The error messages for `GetSecurityInfo` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a `SecurityInputRecord` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `SecurityInfo` IndexedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type should be `SecurityInputRecord.`
FN_IS_RA_20358	The key 'object_name' must be a String object	Specified object_name is not a String object (object_name object should be of type java.lang.String).
FN_IS_RA_20359	The key 'object_id' must be an Integer object	Specified object_id is not an Integer object (object_id object should be of type java.lang.Integer).
FN_IS_RA_11002	<92,2,8> The user, group, or device object information could not be found	The specified object name or object id does not exist on IS.

GetDocClassDesc

GetDocClassDesc is used by the application component to retrieve the description, either of the specified doc class or of all the doc classes. If the client specifies the doc class name or ID as the input parameter, then the description of that doc class is returned. Otherwise, the description of all doc classes is returned.

The Input-Output Records for `GetDocClassDesc` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
DocClassDesc	MappedRecord	DocClassSet	IndexedRecord	The client can also send an empty, unnamed Record instance to retrieve description of all document classes. The output IndexedRecord would hold Map objects corresponding to each document class in the IS.

Description of Input Record `DocClassDesc`:

Key	Value Type	Description	Default Value
doc_class_id	Integer	Document Class Id.	-
doc_class_name	String	The document class name.	-

Description of Map instances contained in Output Record `DocClassSet`:

Key	Value Type	Description	Default Value
doc_class_id	Integer	Document Class Id.	-
doc_class_name	String	The document class name.	-
workspace_name	String	The workspace name of the queue associated with the doc class.	-
queue_name	String	The queue name associated with the doc class.	-
read_access_id	Integer	The read access defined on the doc class in terms of user or group ID.	-
write_access_id	Integer	The write access defined on the doc class in terms of user or group ID.	-
execute_access_id	Integer	The execute access defined on the doc class in terms of user or group ID.	-
pages_per_doc	Integer	The max number of pages in a document of this doc class.	-

The code sample to retrieve the document class description:

```
import java.util.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetDocClassDesc");
//Create the input MappedRecord with name “DocClassDesc”
MappedRecord input =
recordFactory.createMappedRecord("DocClassDesc");
```

```
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
//Iterate through the output IndexedRecord
ListIterator iterator = output.listIterator();
while(iterator.hasNext()){
/* Retrieve each Map object and display the document
class details */
      Map docClassDescMap = (Map) iterator.next();
      System.out.println("DocClassID:" +
docClassDescMap.get("doc_class_id"));
      System.out.println("DocClassName:" +
docClassDescMap.get("doc_class_name"));
      System.out.println("Workspace Name:" +
docClassDescMap.get("workspace_name"));
      System.out.println("Queue Name:" +
docClassDescMap.get("queue_name"));
      System.out.println("Read Access ID:" +
docClassDescMap.get("read_access_id"));
      System.out.println("Write Access ID:" +
docClassDescMap.get("write_access_id"));
      System.out.println("Execute Access ID:" +
docClassDescMap.get("execute_access_id"));
      System.out.println("Pages Per Doc:" +
docClassDescMap.get("pages_per_doc"));
      System.out.println("=============");
}
}catch (ResourceException e) {
       e.printStackTrace();
}
```

The error messages for `GetDocClassDesc` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a `DocClassDesc` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `DocClassSet` IndexedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type should be `DocClassDesc`.

Error Code	Error Message	Description
FN_IS_RA_20361	The key 'doc_class_id' must be an Integer object	Specified doc_class_id is not an Integer object (doc_class_id object should be of type java.lang.Integer).
FN_IS_RA_20360	The key 'doc_class_name' must be a String object	Specified doc_class_ name is not a String object (doc_class_name object should be of type java.lang.String).
FN_IS_RA_11035	Specified document class does not exist	The specified document class does not exist on IS.

GetMenuDesc

GetMenuDesc is used by the application component to retrieve menu description of an index of type menu. It returns the list of all menu items and values for the specified index. If specified index is not of type menu, an exception will be thrown.

The Input-Output Records for `GetMenuDesc` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
IndexName	MappedRecord	MenuItemSet	IndexedRecord	The input contains the menu type index name. The output IndexedRecord contains Map instances that contain menu label and menu value.

Description of the Input Record `IndexName`:

Key	Value Type	Description	Default Value
index_name*	String	The index name of type menu whose menu items and labels are desired	-

MenuItemSet

This is an Indexed Record of Map instances.

Description of Map instances contained in Output Record `MenuItemSet`:

Key	Value Type	Description	Default Value
menu_label	String	Name of menu	-
menu_value	Integer	Corresponding integer value for the menu label.	-

The code sample to get the menu value and menu label name is:

```
import java.util.*;

try  {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetMenuDesc");
```

```
//Create the input MappedRecord with name “IndexName”
MappedRecord input =
recordFactory.createMappedRecord("IndexName");
//Specify the menu type index name
input.put("index_name","Color");
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* Extract the first element of the indexed record if
only one index name is specified. If no index name is
specified in the input record, then the output record
will be a collection of Map objects for each menu type
index */
Map menuItem = (Map) output.get(0);
System.out.println("Menu label:" +
menuItem.get("menu_label"));
System.out.println("Menu value:" +
menuItem.get("menu_value"));
}catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `GetMenuDesc` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not an `IndexName` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `MenuItemSet` IndexedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type should be `DocClassDesc`.
FN_IS_RA_20366	The key 'index_name' is null	Index_name cannot be null as it is a mandatory parameter in `GetMenuDesc` interaction.
FN_IS_RA_10394	Invalid index name	The index name is either null, or not an object of type `java.lang.String` or blank string.

GetCacheList

GetCacheList is used by the application component to retrieve list of all the caches configured on the IS server

The Input-Output Records for `GetCacheList` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
GetCacheListInput	MappedRecord	CacheList	IndexedRecord	The output is an Indexed Record containing the list of cache names configured on IS

Description of the Input Record GetCacheListInput:

Key	Value Type	Description	Default Value
-	-	-	-

CacheList

The output is an Indexed Record containing the list of cache names configured on IS.

Key	Value Type	Description	Default Value
-	-	-	-

The code sample to get the cache list is:

```
import java.util.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetCacheList");
//Create the input MappedRecord with name
“GetCacheListInput”
MappedRecord input =
recordFactory.createMappedRecord("GetCacheListInput");


IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* The output record will be contain a list of all the
caches .Iterate through the output record to retrieve
individual cache names */
List cacheList = new ArrayList();
Iterator iterator = output.iterator();
while(iterator.hasNext()){
cacheList.add(iterator.next());
}
}catch (ResourceException e) {
```

```
            e.printStackTrace();
        }
```

The error messages for `GetCacheList` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a GetCacheListInput MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `CacheList` IndexedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type should be GetCacheListInput.
FN_IS_RA_10378	IndexedRecord name is incorrect	The name of the input Record type should be CacheList.

Note: The IS metadata is refreshed periodically by ISRA. If the application component caches any metadata on its own, then it will have to refresh the data to avoid any data loss or inconsistency.

Password Interactions

GetPasswordStatus

This interaction will enable the client to get the password status of the specified user. The client will specify the user name for which the password status is to be obtained. The input values would be specified through the MappedRecord and the output would also be a MappedRecord.

The execute () method in the Interaction object would call the getPwdStatus() method on ISInterface layer. The ISInterface layer would call the getPasswordStatus() method of the Service Layer class, FN_IS_RPC_SEC_Service, to complete the Interaction.

Note: The Password expiration time may be set only through Default Security Settings option in the IS. To set the password expiration time, two attributes, namely Password Renewal Period and Password Grace Period, need to be set. The settings of these two attributes apply to all users existing on the IS, except the super user. If the two attributes are not set, the password expiration time received through this interaction corresponds to the FileNet IS Base Date.

The Input-Output Records for `GetPasswordStatus` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
StatusRequest	MappedRecord	StatusDescription	MappedRecord	

Description of the Input Record `StatusRequest`:

Key	Value Type	Description	Default Value
object_name	String	Name of user	

Description of MappedRecord instances contained in output record `StatusDescription`:

Key	Value Type	Description	Default Value
nbrLogons	Long	Number of times current user logged in	
successLogTime	Date	Last successful login time	
successWhere	String	Where last successful logon occurred	
failedLogTime	Date	Last failed logon time	
failedWhere	String	Where last failed logon occurred	
failedError	Long	Last failed logon error	
pwdGracePeriod	Boolean	Has the user's password grace period begun	
pwdExpiresTime	Date	Time when password expires	

The code sample to get the password status of the specified user is:

```
import java.util.Date;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetPasswordStatus");
//Create the input MappedRecord with name "StatusRequest"
MappedRecord input =
recordFactory.createMappedRecord("StatusRequest");
//Specify the username
input.put("object_name","User");
//Execute the interaction
MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
/* Extract the elements of the mapped record if and only
if the interaction is a success. If no user name is
specified in the input record, then an error will be
thrown by the IS*/
String numberLogon = (String)
outputRecord.get("nbrLogons");
Date successTime = (Date)
outputRecord.get("successLogTime");
```

```
String successPlace = (String)
outputRecord.get("successWhere");

DatefailedTime = (Date)
outputRecord.get("failedLogTime");

String failedPlace = (String)
outputRecord.get("failedWhere");

String failedErr = (String)
outputRecord.get("failedError");

Boolean gracePeriod = (Boolean)
outputRecord.get("pwdGracePeriod");

Date expirationTime = (Date)
outputRecord.get("pwdExpiresTime");


System.out.println ("Number of Logons:" + numberLogon);

System.out.println("Last Successful Login Time:" +
successTime);

System.out.println("Last successful Login Where:" +
successPlace);

System.out.println("Last Login failure Time:" +
failedTime);

System.out.println("Last Login failure Where:" +
failedPlace);

System.out.println("Last Login failure Error:" +
failedErr);

System.out.println("Grace Period Begun:" + gracePeriod);

System.out.println("Password Expiration Time:" +
expirationTime);

}catch (ResourceException e) {

      e.printStackTrace();

}
```

The error messages for `GetPasswordStatus` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a `StatusRequest` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `StatusDescription` MappedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type and output Record type should be `StatusRequest` and `StatusDescription`, respectively.

Error Code	Error Message	Description
FN_IS_RA_10432	<92,2,8>The user, group, or device object information could not be found.	The user name does not exist on the IS.

ChangePassword

This interaction will enable the client to change the password of a user existing in the IS. The client will specify the old password, new password and the user id for which the password is to be changed. The input values would be specified through the MappedRecord and the output would also be a MappedRecord.

The execute () method in the Interaction object would call the changePassword() method on ISInterface layer. The ISInterface layer would call the changePassword() method of the Service Layer class, FN_IS_RPC_SEC_Service, to complete the Interaction.

Note: After the logged in user (say, u1) changes his/her password, he/she will be able to access the existing session for the same user with the old password (say, p1) for the period of time till the application server destroys the previously managed connection.

This is because application server calls the matchManagedConnection() method on the implemented ManagedConnectionFactory class and uses username and password to validate for existing connections of the user before creating any new connection. The time interval of pool maintenance thread is a configurable parameter through deployment descriptor and can be set to a low value so as to minimize this overlapping time period.

Note: For a user, say user1, to change another user's password, say user2, user1 must have the same administrative group as user2 and in addition user1 must have supervisor rights for that administrative group

If the above rights are set for user1 then he will be able to change the password of that user regardless of user2's old password.

Only SysAdmin has these rights in the IS, by default. For other users to have these rights, SysAdmin has to grant these rights to them.

The Input-Output Records for `ChangePassword` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
PasswordChange	MappedRecord	GenericResult	MappedRecord	

Description of Input Record `PasswordChange`:

Key	Value Type	Description	Default Value
old_password	String	unencrypted old password	
new_password	String	unencrypted new password	
confirm_password	String	confirmation of new password	

Key	Value Type	Description	Default Value
object_name	String	User name whose password is being changed	

Description of MappedRecord instance of Output Record `GenericResult`:

Key	Value Type	Description	Default Value
result	Long	Result Code, perhaps the FileNet error_type	

The code sample to change the password of the user already logged in to IS is:

```
try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("ChangePassword");
//Create the input MappedRecord with name
“PasswordChange”
MappedRecord input =
recordFactory.createMappedRecord("PasswordChange");
//Specify the username, old password, new password and
confirm password
input.put("old_password","abcd");
input.put("new_password","abcdefgh");
input.put("confirm_password","abcdefgh");
input.put("object_name","User");
//Execute the interaction
MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
/* Extract the element of mapped record to check whether
the interaction is a success. If no user name is
specified in the input record, then an error will be
thrown by the IS*/
Long result = (Long)output.get("result");
      long errorTuple = result.longValue();
      if(errorTuple == 0){
            System.out.println("The password has been
changed");
      }else{
            System.out.println("Error in changing the
password. The error tuple is:" + errorTuple);
}
}catch (ResourceException e) {
```

```
            e.printStackTrace();
        }
```

The error messages for `ChangePassword` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a `PasswordChange` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `GenericResult` MappedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type and output Record type should be `PasswordChange and GenericResult,` respectively.
FN_IS_RA_11073	<92,0,221>The length of the password provided is out of range. A password should at least have a minimum length of that specified by the system defaults and cannot have a length greater than 8 characters.	The length of the password is either less than the minimum password length set in the IS or it is greater than 8 characters, which is the maximum password length set by the IS.
FN_IS_RA_11073	<92,2,8>The user, group, or device object information could not be found.	The user name does not exist on the IS.
FN_IS_RA_11073	<92,2,2>The password provided does not match that in the database.	The old password entered by the user is not correct.
FN_IS_RA_10434	The new password is same as the old password. Please re-enter the passwords.	The user's new password entered must be different from the existing password.
FN_IS_RA_10430	New Passwords Mismatch. Please re-enter the new password correctly.	Values entered for new_password and confirm_password are not the same.
FN_IS_RA_11073	<92,0,186>The password update is denied due to inadequate permissions.	The logged in user does not have permissions to change the password of another user.

Print and Fax Interactions

PrintDocs

This interaction will allow users to print (or fax) the documents.

The execute () method in the Interaction object would call the printDocs () method on ISInterface layer. The ISInterface layer would call the printDocs () method of the Service Layer class, FN_IS_RPC_PRI_Service to complete the Interaction.

The Input-Output Records for `PrintDocs` are explained in the following table:

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
PrintRequest	MappedRecord	RequestID	MappedRecord	

Description of Input Record `PrintRequest`:

Key	Value Type	Description	Default Value
DocSpecificationList	Indexed Record	List of document details. Each entry in this list is a Mapped Record of Doc_details	
OptionList	Indexed Record	List of print options associated with each document. Each entry in this list contains a Mapped Record of type Print_Option_Details	
FaxRequest	Boolean	True, if document is to be faxed than printed.	
Notify	Integer	Value for synchronous and asynchronous calls.	1 is asynchronous, 2 is none, 3 is synchronous. Only value supported by ISRA is 2.

DocSpecificationList

This Indexed Record contains another Indexed Record called `DocDetailsIndex`, which in turn contains Mapped Record `DocSpecificationList` for each Document to be printed.

This contains the details of the document to be printed along with page range to be printed.

Key	Value Type	Description	Default Value
doc_id	Long	Document id to be printed or faxed	
FirstPage	Integer	First page of document to be printed.	
LastPage	Integer	Last page of document to be printed.	
ServiceSourceChoice	Integer	Service from which to get documents to be printed.	1 is for Document Service and 2 is for Cache Service. The only value supported by ISRA is 1.

OptionList

This Indexed Record contains another Indexed Record called `PrintOptionsIndex`, which in turn contains Mapped Record `PrintOptionDetails` listing the print properties.

This contains the various print options associated with each document.

Key	Value Type	Description	Default Value
Paper	Short	Paper size.	
Priority	Short	Priority of the print request.	0 is hold. 8 is highest priority. Default is 4.
PrinterName	String	Name of the printer.	
Collate	Boolean	True, if pages are to be collated	False
AccessRestrictions	String	Security access restrictions.	

Key	Value Type	Description	Default Value
Copies	Integer	No. of copies to be printed.	1
PrintTime	Date	Delays print (or fax) request until specified time.	Default is current time.
Staple	Boolean	If true, job will be stapled. Default is false. Not valid for fax requests.	False
TwoSided	Boolean	It true, request will be printed on both sides. Default is false. Not valid for fax requests.	False
Form	String	The name of the form on which to print the request.	
Note	String	Note string is printed on the request header page. Default is no note	
Annotations	Boolean	If true, annotations will be printed. Default is false.	False
RequestHeader	Boolean	If true, print (or fax) a cover page for the request.	False
DocHeader	Boolean	If true, print (or fax) a cover page for each doc of request.	False
Scaling	Short	The scaling operation to be performed.	
Orientation	Short	The rotation to be performed. Not valid for fax requests.	
Overlay	Short	Message to be overlaid on printed document.	2
overlay_units	String	Specifies whether the properties OverlayX and OverlayY are in inches or centimeters	“inches”
overlay_text	String	specifies the overlay text for a print job, that to be printed on documeent	
overlay_xPos	String	Specifies the distance from the left edge where the overlay text begins	
overlay_yPos	String	Specifies the distance from the top edge where the overlay text begins	
EjectTrays	Short	Number of the Eject tray from which to use paper for printing.	
PhoneNumber	String	For fax requests only. Must be specified for all fax requests. No default value.	
HeadlineMessage	String	For fax requests only. This is a required field and the field can not be null.This string will be printed at the top of the each page faxed.	
FaxMode	Integer	For fax requests only.	
PageFootnote	Boolean	For fax requests only. If true, the page number will be overlaid at the bottom of page faxed. Default is false.	False

Key	Value Type	Description	Default Value
TimeFootnote	Boolean	For fax requests only. If true, the time of transmittal will be overlaid at the bottom of each page faxed. Default is false.	False
PhoneExtn	String	For fax requests only. Optional parameter for fax requests. No default value.	
Memo	String	For fax requests only. Optional parameter for fax requests. No default value.	
CoverDoc	Long	For fax requests only. Optional parameter for fax requests. No default value.	
CoverSsn	Long	For fax requests only. Optional parameter for fax requests. No default value.	
PrintService	String	Name of the print service running on IS obtained by executing the interaction “GetPrinterAttributes”.	

Description of Output Record `RequestID`:

Key	Value Type	Description	Default Value
result	long	Request Id for this print request.	

The code sample to print (or fax) the documents is:

```
try{
//Specify the function name for the interaction
interactionSpec.setFunctionName("PrintDocs");
//Create the input MappedRecord with name “PrintRequest”
MappedRecord input =
recordFactory.createMappedRecord("PrintRequest");


//Create an input Indexed Record for document details
with name “DocDetailsIndex”
IndexedRecord docDetailsIndex =
recordFactory.createMappedRecord ("IndexedRecord",
“DocDetailsIndex”);
//Create an input Mapped Record for each documents’
details with name “DocSpecificationList”
MappedRecord doc_details =
recordFactory.createMappedRecord ("MappedRecord",
"DocSpecificationList");


doc_details.put("doc_id", new Long(123456));
```

```
doc_details.put("FirstPage", new Integer(1));
doc_details.put("LastPage", new Integer(4));
doc_details.put("ServiceSourceChoice", new Integer(1));
docDetailsIndex.add(doc_details);

//Create an input Indexed Record for document details
with name “PrintOptionsIndex”
IndexedRecord printOptionsIndex =
recordFactory.createMappedRecord
("IndexedRecord","PrintOptionsIndex");
//Create an input Indexed Record for document details
with name “PrintOptionDetails”
MappedRecord print_option_details =
recordFactory.createMappedRecord
("MappedRecord","PrintOptionDetails");

print_option_details.put("Paper",new Short((short)11));
print_option_details.put("Priority",new Short((short)4));
print_option_details.put("PrinterName",
"FileNetPrinter");
print_option_details.put("Collate", new Boolean(false));
print_option_details.put("AccessRestrictions", new
Boolean(false));
print_option_details.put("Copies", new Integer(1));
print_option_details.put("Staple", new Boolean(false));
print_option_details.put("TwoSided", new Boolean(false));
print_option_details.put("Form", "This is a form
example");
print_option_details.put("Note", "This is a note
example");
//"PrintTime" is an optional parameter for both Print and
Fax requests
print_option_details.put("PrintTime", new
java.util.Date());
//"Annotations" is an optional parameter for both Print
and Fax requests
print_option_details.put("Annotations", new
Boolean(false));
```

```
//"RequestHeader" is an optional parameter for both Print
and Fax requests
print_option_details.put("RequestHeader", new
Boolean(true));
print_option_details.put("DocHeader"  , new
Boolean(false));
print_option_details.put("Scaling", new Short((short)0));
print_option_details.put("Orientation", new
Short((short)0));
print_option_details.put("Overlay", new Short((short)2));
print_option_details.put("overlay_text", new String(“This
is Overlay Text”));
print_option_details.put("overlay_units", new
String(“I”));
print_option_details.put("overlay_xPos", new
String(“2”));
print_option_details.put("overlay_yPos", new
String(“2”));


print_option_details.put("EjectTrays", new
Short((short)0));
//All the following parameters are valid only for Fax
requests
print_option_details.put("PhoneNumber", "1234-4567-
7890");
print_option_details.put("HeadlineMessage" ,"This is a
test Headline Message");
print_option_details.put("FaxMode", new Short((short)0));
print_option_details.put("PageFootnote", new
Boolean(false));
print_option_details.put("TimeFootnote", new
Boolean(false));
print_option_details.put("PhoneExtn", "9876");
print_option_details.put("Memo", "This is a memo
example");
print_option_details.put("CoverDoc", new Long(123321));
print_option_details.put("CoverSsn", new Long(123));
print_option_details.put(“PrintService”, “PrintServer”);
printOptionsIndex.add(print_option_details);
```

```
input.put("DocSpecificationList" , docDetailsIndex);
input.put("OptionList", printOptionsIndex);
input.put("FaxRequest" , new Boolean(true));
input.put("Notify", new Short((short)2));

//Execute the interaction
MappedRecord output = (MappedRecord)
interaction.execute(interactionSpec,input);
/* Extract the elements of the mapped record if and only
if the interaction is a success.*/
long RequestId =
((Long)output.get("result")).longValue();
System.out.println("Request Id for Print/Fax request:" +
RequestId);
}catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `PrintDocs` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a `PrintRequest` MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `RequestID` MappedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the input Record type and output Record type should be `PrintRequest and RequestID`, respectively.
FN_IS_RA_10435	Priority should be between 1 and 9.	-
FN_IS_RA_10436	No. of copies should be between 1 and 100	-
FN_IS_RA_10437	Scaling should be between 0 and 6	-
FN_IS_RA_10438	Orientation should be between 0 and 3	-
FN_IS_RA_10439	Fax Mode should be between 0 and 5	-
FN_IS_RA_10440	Phone number is required field for fax requests	-
FN_IS_RA_10441	First Page should be of Integer type	-
FN_IS_RA_10442	Last Page should be of Integer type	-
FN_IS_RA_10443	Paper type should be of Short type	-
FN_IS_RA_10444	Priority should be of Short type	-
FN_IS_RA_10445	Printer Name should be String	-

Error Code	Error Message	Description
FN_IS_RA_10446	Collate should be of Boolean type	-
FN_IS_RA_10447	Copies should be of Integer type	-
FN_IS_RA_10448	Print Time should be of Date type	-
FN_IS_RA_10449	Staple should be of Boolean type	-
FN_IS_RA_10464	Form should be of String type	-
FN_IS_RA_10465	Note should be of String type	-
FN_IS_RA_10466	Annotation should be of Boolean type	-
FN_IS_RA_10467	Request Header should be of Boolean type	-
FN_IS_RA_10468	Doc Header should be of Boolean type	-
FN_IS_RA_10469	Scaling should be of Short type	-
FN_IS_RA_10470	Orientation should be of Short type	-
FN_IS_RA_10471	Phone Number should be of String type	-
FN_IS_RA_10472	Headline Message should be of String type	-
FN_IS_RA_10473	Fax Mode should be of Short type	-
FN_IS_RA_10474	Page Footnote should be of Boolean type	-
FN_IS_RA_10475	Time Footnote should be of Boolean type.	-
FN_IS_RA_10476	Printer name is invalid	-
FN_IS_RA_10477	This paper type is not supported by specified printer/fax	-
FN_IS_RA_10478	This option is not valid for fax request	-
FN_IS_RA_10479	This option is not valid for print request.	-
FN_IS_RA_10480	Overlay option should be of Short type.	-
FN_IS_RA_10481	Eject Tray option should be of Short type.	-
FN_IS_RA_10482	Phone Extension option should be of String type	-
FN_IS_RA_10483	Memo option should be of String type	-
FN_IS_RA_10484	CoverDoc option should be of Long type	-
FN_IS_RA_10485	CoverSsn option should be of Long type	-
FN_IS_RA_10486	Overlay should be between 0 and 2	-
FN_IS_RA_10487	Eject Tray should be between 0 and 15	-
FN_IS_RA_10488	Note length cannot be more than 42 characters	-
FN_IS_RA_10489	Form length cannot be more than 32 characters	-
FN_IS_RA_10490	Phone Number length cannot be more than 40 characters	-
FN_IS_RA_10491	Headline Message length cannot be more	-

Error Code	Error Message	Description
	than 98 characters	
FN_IS_RA_10492	Memo length cannot be more than 376 characters	-
FN_IS_RA_10493	Phone Extn. length cannot be more than 48 characters	-
FN_IS_RA_10494	Collate is not a valid option for fax request	-
FN_IS_RA_10495	Copies is not a valid option for fax request	-
FN_IS_RA_10496	TwoSided is not a valid option for fax request	-
FN_IS_RA_10497	Staple is not a valid option for fax request	-
FN_IS_RA_10498	DocHeader is not a valid option for fax request	-
FN_IS_RA_10499	Scaling is not a valid option for fax request	-
FN_IS_RA_10500	Orientation is not a valid option for fax request	-
FN_IS_RA_10504	EjectTrays is not a valid option for fax request	-
FN_IS_RA_10505	Phone Number is not a valid option for print request	-
FN_IS_RA_10506	Headline Message is not a valid option for print request	-
FN_IS_RA_10507	Fax Mode is not a valid option for print request	-
FN_IS_RA_10508	Page Footnote is not a valid option for print request	-
FN_IS_RA_10509	Time Footnote is not a valid option for print request	-
FN_IS_RA_10510	Memo is not a valid option for print request	-
FN_IS_RA_10511	Phone Number Extension is not a valid option for print request	-
FN_IS_RA_10512	CoverDoc is not a valid option for print request	-
FN_IS_RA_10513	CoverSsn is not a valid option for print request	-
FN_IS_RA_10514	Invalid Document details Object Type	-
FN_IS_RA_10515	Invalid Print Options Object Type	-
FN_IS_RA_10516	Invalid printer type object	-
FN_IS_RA_10517	Error Occurred in interaction Print/Fax documents	-
FN_IS_RA_10518	Selected printer can't staple	-
FN_IS_RA_10519	Selected printer can't print two-sided	-
FN_IS_RA_10520	Form is not a valid option for fax request	-

Error Code	Error Message	Description
FN_IS_RA_10521	Note is not a valid option for fax request	-
FN_IS_RA_10522	Overlay is not a valid option for fax request	-
FN_IS_RA_10523	Print/Fax Time can't be less than current time.	-
FN_IS_RA_10524	Headline Message field can't be empty	-
FN_IS_RA_10525	Fax Name is invalid.	-
FN_IS_RA_10526	There is no available printer to satisfy the specified print options	-
FN_IS_RA_10527	There is no available fax machine to satisfy the specified fax options.	-
FN_IS_RA_10531	Overlay text should be of String type	-
FN_IS_RA_10532	Overlay units should be of String type ("C" for Centimeters "I" for Inches)	-
FN_IS_RA_10533	Overlay positions should be of String type	-
FN_IS_RA_10517	<90,0,6>Requested record not found	-
FN_IS_RA_40031	Invalid page no. specified	-
FN_IS_RA_10517	<87,1,21>Invalid #of pages in request. A request with no overlay option must have at least one page, and a request with the overlay option must have at least two pages.	Error received when overlay option is specified for single page document printing.
FN_IS_RA_40087	Print Service Name can not be null.	
FN_IS_RA_40088	Print Service Name should be String.	

GetPrinterAttributes

This interaction will allow users to get the printer attributes for the printers configured on IS. As no data is required from the client, the client will send an empty anonymous Record instance, i.e., an instance of Record without any specific name and data. The output record would be an IndexedRecord called PrinterAttributes, which would contain list of Mapped records of printer attributes.

The execute () method in the Interaction object would call the getServiceAttributes () method on ISInterface layer. The ISInterface layer would call the getServiceAttributes () method of the Service Layer class, FN_IS_RPC_PRI_Service to complete the Interaction.

Input Record Description		Output Record Description		Remarks
Name	**Type**	**Name**	**Type**	
PrinterAttributes	MappedRecord	PrinterAttribs	Indexed Record	The input record is null or an empty Record object. The output IndexedRecord will contain list of printer attribs

				objects.

Description of Output Record `PrinterAttributes`:

Key	Value Type	Description	Default Value
-	Indexed Record	List of printer attributes. Each entry in this list is a Mapped Record of printer_attribs	

Printer_Attribs

This contains the printer attributes of printer.

Key	Value Type	Description	Default Value
PrinterName	String	Name of the printer	
Fax	Boolean	True, if it is a fax machine.	
Security	String	Security attributes	
Speed	Integer	Speed of the printer in pages/min	
TwoSided	Boolean	True, if printer can print on both sides.	
Staple	Boolean	True, if printer has stapler.	
Trays	Integer	No. of trays in this printer	
PaperSupported	Vector	Vector of paper sizes, this printer can support.	
PrintService	String	Name of the print service running on IS.	

The code sample to get the printer attributes for the printers configured on IS is:

```
import java.util.*;


try {
//Specify the function name for the interaction
interactionSpec.setFunctionName("GetPrinterAttributes");
//Create the input MappedRecord with name
“PrinterAttributes”
MappedRecord input =
recordFactory.createMappedRecord("PrinterAttributes");
//Execute the interaction
IndexedRecord output = (IndexedRecord)
interaction.execute(interactionSpec,input);
/* Extract the elements of the indexed record if and only
if the interaction is a success.*/
for(int i = 0; i <output.size(); i++){
HashMap priA = (HashMap)output.get(i);
```

```
String printerName = (String) priA.get("PrinterName");
Integer fax = (Integer) priA.get("Fax");
String security = (String) priA.get("Security");
Integer speed = (Integer) priA.get("Speed");
Boolean twoSided = (Boolean) priA.get("TwoSided");
Boolean staple = (Boolean) priA.get("Staple");
Integer trays = (Integer) priA.get("Trays");
String paperSupported = (String)
priA.get("PaperSupported");
String printService = (String) priA.get("PrintService");

System.out.println ("Name of Printer:" + printerName);
System.out.println("Is Fax:" + fax);
System.out.println("Security Permissions:" + security);
System.out.println("Speed:" + speed);
System.out.println("Two Sided option:" + twoSided);
System.out.println("Staple option:" + staple);
System.out.println("Number of Eject Trays:" + trays);
System.out.println("Papers Supported:" + paperSupported);
System.out.println("print Service:" + printService);

}
}catch (ResourceException e) {
      e.printStackTrace();
}
```

The error messages for `GetPrinterAttributes` interaction are listed in the following table:

Error Code	Error Message	Description
FN_IS_RA_10354	Input Record type is invalid	The input Record type is not a MappedRecord.
FN_IS_RA_10373	Output Record type is invalid	The output Record type is not a `PrinterAttributes` IndexedRecord.
FN_IS_RA_10358	MappedRecord name is incorrect	The name of the output Record type should be `PrinterAttributes`.

Appendix A: References

This section describes the classes and their methods that can be used by an application component. This section contains a sample code, which shows how to bind a `ConnectionFactory` object into the name space of a JNDI service.

The classes covered here are:

- com.filenet.is.ra.cci.FN_IS_CciInteractionSpec.
- com.filenet.is.ra.cci.FN_IS_CciConnectionSpec.

Note: The classes `FN_IS_CciInteractionSpec` and `FN_IS_CciConnectionSpec` are provided in ISRA.jar as well as client_helper.jar. In a Managed Environment, client_helper.jar needs to be included in the Application Server's CLASSPATH.

This section also contains:

- GetDocProperties in View Edition.
- Queue Field type description.

Class FN_IS_CciInteractionSpec

This class resides in `com.filenet.is.ra.cci` package and inherits `java.lang.Object` class.

The constructors supported by the class FN_IS_CciInteractionSpec() are described in the following table:

Constructor Name	Description
FN_IS_CciInteractionSpec()	Takes no parameters.

The methods supported by the class FN_IS_CciInteractionSpec() are described in the following table:

Return Type	Method Name	Description
int	getExecutionTimeout()	Gets the execution timeout in seconds for the interaction.
int	getFetchDirection()	Gets the FetchDirection for the ResultSet returned by this interaction.
int	getFetchSize()	Gets the FetchSize for the ResultSet returned by this interaction.
String	getFunctionName()	Gets the FunctionName for the interaction to be executed.
int	getInteractionVerb()	Gets the InteractionVerb for the interaction to be executed.

Return Type	Method Name	Description
int	getResultSetConcurrency()	Gets the ResultSetConcurrency for the ResultSet returned by this interaction.
int	getResultSetType()	Gets the ResultSetType for the ResultSet returned by this interaction.
javax.resource.cci.ResourceWarning	getWarning()	Gets the ResourceWarning generated by this instance.
void	setExecutionTimeout (int iTimeOut)	Sets the timeout for the interaction.
void	setFetchDirection(int iFetchDirection)	Sets the FetchDirection for the ResultSet returned by this interaction.
void	setFetchSize(int iFetchSize)	Sets the FetchSize for the ResultSet returned by this interaction.
void	setFunctionName (String functionName)	Sets the FunctionName for the interaction to be executed.
void	setInteractionVerb(int iVerb)	Sets the InteractionVerb for the interaction to be executed.
void	setResultSetConcurrency(int iResultSetConcurrency)	Sets the ResultSetConcurrency for the ResultSet returned by this interaction.
void	setResultSetType(int iResultSetType)	Sets the ResultSetType for the ResultSet returned by this interaction.

The methods inherited in the class FN_IS_CciInteractionSpec() from the class `java.lang.Object` are listed in the following table:

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Class FN_IS_CciConnectionSpec

This class resides in `com.filenet.is.ra.cci` package and inherits `java.lang.Object` class.

The constructors supported by the class FN_IS_CciConnectionSpec() are described in the following table:

Constructor Name	Description
FN_IS_CciConnectionSpec()	Constructs an empty FN_IS_CciConnectionSpec instance.
FN_IS_CciConnectionSpec(String userID, String passwd)	Constructs FN_IS_CciConnectionSpec instance with specified userName and password.
FN_IS_CciConnectionSpec(String userID, String passwd , Integer loginType)	Constructs FN_IS_CciConnectionSpec instance with specified username, password and loginType.If loginType is 0 then direct IS login procedure is initiated. And if loginType is 1 then the username and password

Constructor Name	Description
	are authenticated on the LDAP Server mentioned in the Deployment Descriptor.

The methods supported by the class FN_IS_CciConnectionSpec()are described in the following table:

Return Type	Method Name	Description
String	getPassword()	Returns password to caller.
String	getUserName()	Returns userID to caller.
Integer	getLoginType()	Returns LoginType to caller.
Void	setUserName(String userID)	Sets the username.
Void	setPassword(String password)	Sets the password.
Void	setLoginType(Integer loginType)	Sets the login Type

The methods inherited in the class FN_IS_CciConnectionSpec() from the class java.lang.Object are listed in the following table:

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Binding ConnectionFactory into the JNDI Namespace

The code sample given below can be used to bind a reference of a ConnectionFactory to a JNDI namespace; this code will be useful in the Non-Managed (with JNDI) scenario:

```
// Create a ManagedConnectionFactory instance:

FN_IS_SpiManagedConnectionFactory
managedconnectionFactory = new

FN_IS_SpiManagedConnectionFactory();

managedconnectionFactory.setDomainName("IS_Domain"); //
DomainName of IS Server.

managedconnectionFactory.setOrganizationName("FileNet");//
/organization name of IS.

managedconnectionFactory.setProductName("FileNet ISRA
");// product name.

managedconnectionFactory.setLogFileName(“ISRA.log”);//Log
filename.

managedconnectionFactory.setLoggingLevel(new Integer(2));

managedconnectionFactory.setLoggingMode(new Integer(2));

managedconnectionFactory.setPageBufferSize(new
Integer(64));
```

```
managedconnectionFactory. SetLdapImplClassName
(“com.filenet.is.ra.fnis.FN_IS_IPlanetImpl”);

managedconnectionFactory. SetLdapImplClassString
(“filenetserver;389;ou=filenet,ou=people,dc=filenetIS”);

managedconnectionFactory.setInherentLogin (new
Integer(1));

//Create a connection factory using the default
connection manager.

Object cf =
managedconnectionFactory.createConnectionFactory(new

FN_IS_SpiConnectionManager());

ByteArrayOutputStream baos = new ByteArrayOutputStream();

 try {

     ObjectOutputStream oos = new
ObjectOutputStream(baos);

     oos.writeObject(cf);

     oos.flush();

     }

 catch (IOException ioe) {

 System.out.println("caught IOException

     writing connection factory to output stream");

 throw ioe;

 }

 /* Create a Reference object containing the connection
factory as a byte array and indicating the class name of
the objectFactory implementation. FN_IS_ObjectFactory
class is part of ISRA, and provides implementation for
getObjectInstance() of javax.naming.spi.ObjectFactory
interface.*/

 String objectFactoryClassName = new String
("com.filenet.is.ra.spi.FN_IS_ObjectFactory");

 ref = new Reference( “ConnectionFactory for ISRA”,

new BinaryRefAddr("ISRACF", baos.toByteArray()),

 objectFactoryClassName, null);

 javax.naming.Context context = getJNDIContext();

 try {

//Bind it to the JNDI using the name “FileNet_ISRA_CF”.

 context.rebind(“FileNet_ISRA_CF”, ref);
```

```
}
catch (NamingException ne) {
System.out.println("caught NamingException doing rebind
of Reference");
throw ne;
}
```

Note: The first parameter (String addrType) to the BinaryRefAddr constructor is “ISRACF”.

FN_IS_SpiManagedConnectionFactory Methods

setLogWriter()

```
public void setLogWriter(PrintWriter logWriter)

throws ResourceException
```

Sets the log writer for this ManagedConnectionFactory instance. The log writer is a character output stream to which all logging and tracing messages for this ManagedConnectionfactory instance would be printed.

setProductVersion()

```
public void setProductVersion(String productVersion)
```

Sets the product version. Product version can be retrieved using the getEISProductVersion() method of ManagedConnectionMetaData .

setProductName()

```
public void setProductName(String productName)
```

Sets the product name. The product name can be retrieved using the getEISProductName () method of ManagedConnectionMetaData.

setDomainName()

```
public void setDomainName(String domainName)
```

Sets the domain name. The domain name and organization are used to identify IS server.

setOrganizationName()

```
public void setOrganizationName(String organizationName)
```

Sets the organization name. The domain name and organization are used to identify IS server.

setVendorName()

```
public void setVendorName(String vendorName)
```

Sets the vendor name. The vendor name can be retrieved using the getAdapterVendorName () method of ResourceAdapterMetaData.

setSpecVersion()

```
public void setSpecVersion(String specVersion)
```

Sets the spec version, which can be retrieved using the getAdapterVersion() method of ResourceAdapterMetaData.

setPageBufferSize()

```
public void setPageBufferSize(java.lang.Integer bufferSize)
```

Sets the buffer size. The buffer size is used in GetDocumentContent interaction, to read the next chunk of data from the underlying stream.

setCacheRefreshInterval ()

```
public void setCacheRefreshInterval(java.lang.Integer
cacheRefreshInterval)
```

Sets the cache refresh interval (in hours). Data, such as document class indexes; menu values, etc. will be cached by ISRA and refreshed periodically. If no value is specified for this property, default value (one hour) will be used to refresh ISRA cache.

setLogFileName()

```
public void setLogFileName(String fileName)
```

Sets the Log file name. The parameter is Log File name with absolute path.

setLogFileSize()

```
public void setLogFileSize(Integer fileSize)
```

Sets the Log file size. The Log File is made in the home directory of the machine where the application server exists.

setLoggingLevel()

```
public void setLoggingLevel(Integer level)
```

Sets the logging level. The level can be 0, 1 or 2.

0 - Errors and warnings.

1 - Informative messages.

2 - Debugging.

setLoggingMode()

```
public void setLoggingMode(Integer mode)
```

Sets the logging mode. Mode can be 0, 1, 2 or 3.

0 - No logging.

1 - Only console logging.

2 - Only file logging.

3 - File as well as console logging.

setLdapImplClassName()

```
public void getLdapImplClassName (String ldapImplClassName)
```

Sets the class name of the implementation class for a specific LDAP server in a private variable.

setLdapImplClassString()

```
public void setLdapImplClassString (String
ldapImplClassString)
```

Sets the LDAP Description string for a specific LDAP server in a private variable.

setInherentLogin()

```
public void setInherentLogin (Integer inherentLogin)
```

Sets the Inherent method that ISRA will use for all logins unless specified by the user.

setDeploymentInstance()

```
public void
setDeploymentInstance(IntegerdeploymentInstance)
```

Sets the value of the instance of ISRA that is being deployed on a single machine.

setRPCLogging()

```
public void setRPCLogging(java.lang.Integer rpcLogging)
```

Sets the logging level for the logging in Java COR Layer. Logging level can be 0, 1 or 2.

0-Exceptions and Warnings.

1-Info.

2-Debug Level.

setPCHLogging()

```
public void setPCHLogging(String pchLogging)
```

Sets the value of the PCH logging that is used to enable or disable pch logging in ISRA.

setAllowAnonymousUser()

```
public void setAllowAnonymousUser (Boolean
allowAnonymousUser)
```

Sets the value of AllowAnonymousUser that is used to enable or disable LDAP Anonymous user login in ISRA.

AddDoc and UpdateDocProperties Interactions

This section contains information about the input parameters passed in the AddDoc and UpdateDocProperties interactions.

Mandatory parameters: If the mandatory document properties are null, an appropriate exception will be thrown in both the interactions.

Optional parameters: If any optional document properties are null, default value of the appropriate index type will be stored while adding a new document, or updating existing document properties.

The table below describes the various index type names, values in the IS and corresponding Java data types.

IS Index Type	Value	Corresponding Java Data Type	Description
BOOLEAN	0x0041	Boolean	True or False.
BYTE	0x0042	Byte	Signed two's compliment 8-bit value. Minimum value is ' –128 ' and maximum value is ' +127 '.
UNSBYTE	0x0043	Short	Unsigned 8-bit value. Minimum value is ' 0 ' and maximum value is ' +255 '.
SHORT	0x0044	Short	Signed two's compliment 16-bit value. Minimum value is ' – 32768 ' and maximum value is ' +32767 '.
UNSSHORT	0x0045	Integer	Unsigned 16-bit value. Minimum value is ' 0 ' and maximum value is ' +65535 '.
LONG	0x0046	Integer	Signed two's compliment 32-bit value. Minimum value is ' –2147483648 ' and maximum value is ' +2147483647 '.
UNSLONG	0x0047	Long	Unsigned 32-bit value. Minimum value is ' 0 ' and maximum value is '4294967295 '.

IS Index Type	Value	Corresponding Java Data Type	Description
FP_NUM	0x0031	BigDecimal	FileNet floating point number. FPNumber minimum value is '.99999999999999E-259', and maximum value is '.99999999999999E+252'. **Note**: User should preferably construct BigDecimal object using constructor that takes 'String' parameter wherever FP_NUM is used.
ASCII	0x0032	String	String data.
DATE	0x0038	Date	IS encoded date.
TIME	0x0033	Date	IS encoded date and time.
MENU	0x0034	Integer	Menu value is Unsigned 16 bit value. Minimum value is ' 0 ' and maximum value is ' +65535'.

The following table describes the various system indexes, index type, and restrictions imposed while updating certain system index values.

Sr. No.	Index Name	IS Index Type	Remarks
1	F_DOCNUMBER	UNSLONG	Cannot be updated by client.
2	F_DOCCLASSNUMBER	UNSSHORT	Cannot be updated by the client.
3	F_ENTRYDATE	DATE	Cannot be updated by the client.
4	F_ANNOTATIONFLAG	BOOLEAN	Cannot be updated by the client.
5	F_ARCHIVEDATE	DATE	Can be updated.
6	F_DELETEDATE	DATE	Can be updated.
7	F_RETENTBASE	UNSBYTE	Can be updated.
8	F_RETENTDISP	UNSBYTE	Can be updated.
9	F_RETENTOFFSET	UNSLONG	Can be updated.
10	F_PAGES	UNSSHORT	Cannot be updated by the client.
11	F_ACCESSRIGHTS	ASCII	Can be updated.
12	F_DOCTYPE	UNSBYTE	Can be updated.
13	F_CLOSED	BOOLEAN	Cannot be updated by the client.
14	F_DOCFORMAT	ASCII	Can be updated.
15	F_DOCLOCATION	ASCII	Can be updated.

Following are the valid values for the system index F_DOCTYPE used in AddDoc and UpdateDocProperties interactions:

Document Type	Value	Description
INX_IMAGE_DOC	0	Image documents
INX_NULL_DOC	48	Obsolete
INX_TEXT_DOC	49	Text/Cold documents without background image
INX_FORM_DOC	50	Legacy form documents generated from WFD software
INX_MIXED_DOC	51	Cold documents with background image or Multipage unknown documents

Document Type	Value	Description
INX_ANNOT_DOC	52	Obsolete
INX_OTHER_DOC	53	Unknown/ Other documents
INX_SEP_SHEET	57	Obsolete

Note: If F_DOCTYPE is not specified in the input record, it will be taken as INX_OTHER_DOC (53).

Following are some valid values for the system index F_DOCFORMAT used in AddDoc and UpdateDocProperties interactions:

Document containing a single page:

F_DOCFORMAT = <page mime type>; name="<page name>"

Document containing multiple pages of the same mime type:

F_DOCFORMAT = <page mime type>

Document containing multiple pages of different mime type:

F_DOCFORMAT = "application/octet-stream"

GetDocProperties in View Edition

The code sample to achieve GetDocProperties in view edition, using FindDocuments and GetDocClassIndices interactions is:

```
// Step 1: Execute FindDocuments interaction to get the
DocClassName

FN_IS_CciInteractionSpec interactionSpec = new
FN_IS_CciInteractionSpec();

interactionSpec.setFunctionName("FindDocuments");

MappedRecord input=
recordFactory.createMappedRecord("QueryRequest");

input.put ("query", "select F_DOCCLASSNAME from
FnDocument where F_DOCNUMBER = 100022");//Specify the
document id in the query

input.put("max_rows", new Integer(1));

ResultSet resultSet = (ResultSet)
interaction.execute(interactionSpec,input);

resultSet.next();

String docClassName =
resultSet.getString("F_DOCCLASSNAME");
```

```
// close the ResultSet and the Interaction objects

resultSet.close();

interaction.close();

// create the interaction object

interaction = connection.createInteraction();

/* Step 2: Execute GetDocClassIndices interaction to get
the list of indices */

interactionSpec.setFunctionName("GetDocClassIndices");

input= recordFactory.createMappedRecord("DocClassName");

input.put("docclass_name",docClassName);

IndexedRecord output =(IndexedRecord)
interaction.execute(interactionSpec, input);

ListIterator iterator = output.listIterator();

// Create a comma separated list of indexNames from the
output record

String commaSeparatedIndexNames = "";

while(iterator.hasNext()){

      Map map = (Map)iterator.next();

      String indexName = (String)map.get("index_name");

      commaSeparatedIndexNames += ("," + indexName);

}

/* Step 3: Execute FindDocuments interaction to get the
values for these indices */

input= recordFactory.createMappedRecord("QueryRequest");

input.put ("query", "select " + commaSeparatedIndexNames
+ " from FnDocument where F_DOCNUMBER = 100022");

input.put("max_rows", new Integer(1));

resultSet = (ResultSet)
interaction.execute(interactionSpec,input);

ResultSetMetaData resultSetMetaData =
resultSet.getMetaData();

int columnCount = resultSetMetaData.getColumnCount();

System.out.println("Total number of indices:" +
columnCount);

//Create a list of column names and another list for
column values

List columnNameList = new ArrayList(columnCount);
```

```
 for(int index = 1; index <= columnCount; index++){

columnNameList.add(resultSetMetaData.getColumnName(index)
);
}
List columnValueList = new ArrayList(columnCount);
resultSet.next();
for(int index = 1; index <= columnCount; index++){
      columnValueList.add(resultSet.getObject(index));
}
//Display both the lists
for(int index = 0; index < columnCount; index++){
     System.out.print(columnNameList.get(index) + ":");
     System.out.print(columnValueList.get(index));
}
```

Queue Field Type Description

The table below describes the various IS queue field data types, and corresponding Java data types that can be used with the GetQueueFields interaction:

Value	IS Queue Field Type	Corresponding Java Data Type
1	Number	BigDecimal
2	String	String
3	Time	Date
4	Selection	String
5	Document	Long
6	Folder	Long
7	Integer	Long
8	Date	Date
10	boolean	Boolean
11	null	null
12.	Decimal	BigDecimal

System Fields in Queue Query

The table below describes the various system fields, which can be used in query with GetQueueEntries interaction:

Serial No	System field name	Description
1	F_TIMEOUT	The date and time after which an item is considered “too old” to still be in this queue. In other words, it should have been processed by then.
2	F_PRIORITY	Priority level for an item in a queue ranges from 0 (lowest) to 9 (highest). The system default (at insertion) is 5.
3	F_GROUP_NAME	The group name associated with the entry.
4	F_USER_NAME	The user name associated with the entry.
5	F_BUSY	A boolean value indicating the status of an item in the queue.

Appendix B: XML Schema for Image Manager Annotations

This section contains the following XSD files:

- FnDocAnnolist.xsd
- FnSecurity.xsd

FnDocAnnoList.xsd

```
<?xml version="1.0" encoding="utf-8"?>

<xsd:schema id="FnDocAnnoList"
targetNamespace="http://tempuri.org/FnDocAnnoList.xsd"
elementFormDefault="qualified"
xmlns="http://tempuri.org/FnDocAnnoList.xsd"
xmlns:sec="http://tempuri.org/FnSecurity.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:NS="http://tempuri.org/FnDocAnnoList.xsd">

    <xsd:import namespace="http://tempuri.org/FnSecurity.xsd"
schemaLocation="C:\Inetpub\wwwroot\IdmWSX\DataProvider\XML\FnSecurity.
xsd" />

    <xsd:element name="FnDocAnnoList" type="FnDocAnnoListType">

    </xsd:element>

    <!--

              Note: The "schemaLocation" may need to be modified to
reflect the users configuration.

              Define a collection of Annotation objects for the
specified page of an IS or a CS document.

       -->

    <xsd:complexType name="FnDocAnnoListType">

        <xsd:sequence>

            <xsd:element name="FnPageAnnoList"
type="FnPageAnnoListType" minOccurs="0" maxOccurs="unbounded" />

            <xsd:element name="FnAnnoDefPermission"
type="FnAnnoDefPermissionType" />

        </xsd:sequence>

        <xsd:attribute name="LibName" type="xsd:string" use="optional"
/>

        <xsd:attribute name="DocID" type="xsd:string" use="optional"
/>
```

```
        <xsd:attribute name="SystemType" type="xsd:string" />
    </xsd:complexType>
    <!--
              Define the annotation collection type.
       -->
    <xsd:complexType name="FnPageAnnoListType">
        <xsd:sequence>
            <xsd:element name="FnAnno" minOccurs="0"
maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="PropDesc"
type="PropDescType" />
                        <xsd:element name="security"
type="sec:securitytype" />
                    </xsd:sequence>
                    <xsd:attribute name="STATE" type="FnAnnoState" />
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="Page" type="xsd:integer" use="required"
/>
    </xsd:complexType>
    <!--
              The simple type definition for the annotation state.
              value=”none” the annotation hasn’t been changed since it
is retrieved.
              value=”change” the annotation has been modified.
              value=”add” a newly created annotation.
              value=”remove” this annotation is marked to be deleted.
       -->
    <xsd:simpleType name="FnAnnoState">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="none" />
            <xsd:enumeration value="change" />
```

```
                <xsd:enumeration value="add" />
                <xsd:enumeration value="remove" />
            </xsd:restriction>
        </xsd:simpleType>
        <!--
                The complex type defines the structure for the F_POINTS
    property. Currently it is not used. ListofFnByte is used instead.
           -->
        <xsd:complexType name="FnPoints">
            <xsd:sequence>
                <xsd:element name="point" minOccurs="0" maxOccurs="300">
                    <xsd:complexType>
                        <xsd:attribute name="X" type="xsd:short" />
                        <xsd:attribute name="Y" type="xsd:short" />
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
        <!--
                 Definition of a byte that has a value from 0 to 255.
           -->
        <xsd:simpleType name="FnByte">
            <xsd:restriction base="xsd:integer">
                <xsd:minInclusive value="0" />
                <xsd:maxInclusive value="255" />
            </xsd:restriction>
        </xsd:simpleType>
        <!--
                This defines a list of bytes to be used by F_POINTS and
    F_CUSTOM_BYTE.
           -->
        <xsd:simpleType name="listOfFnByte">
            <xsd:list itemType="FnByte" />
```

```
    </xsd:simpleType>
    <!--
             This complex type defines the annotation properties.
       -->
    <xsd:complexType name="PropDescType">
        <xsd:sequence>
            <xsd:element name="F_CUSTOM_BYTES" type="listOfFnByte" />
            <xsd:element name="F_POINTS" type="listOfFnByte" />
            <xsd:element name="F_TEXT" type="xsd:string" />
        </xsd:sequence>
        <xsd:attribute name="F_ANNOTATEDID" type="xsd:string" />
        <xsd:attribute name="F_ARROWHEAD_SIZE" type="xsd:short" />
        <xsd:attribute name="F_BACKCOLOR" type="xsd:long" />
        <xsd:attribute name="F_BORDER_BACKMODE" type="xsd:short" />
        <xsd:attribute name="F_BORDER_COLOR" type="xsd:long" />
        <xsd:attribute name="F_BORDER_STYLE" type="xsd:short" />
        <xsd:attribute name="F_BORDER_WIDTH" type="xsd:short" />
        <xsd:attribute name="F_BRUSHCOLOR" type="xsd:long" />
        <xsd:attribute name="F_CLASSNAME" type="xsd:string" />
        <xsd:attribute name="F_CLASSID" type="xsd:string"
use="required" />
        <xsd:attribute name="F_ENTRYDATE" type="xsd:dateTime" />
        <xsd:attribute name="F_FONT_BOLD" type="xsd:boolean" />
        <xsd:attribute name="F_FONT_ITALIC" type="xsd:boolean" />
        <xsd:attribute name="F_FONT_NAME" type="xsd:string" />
        <xsd:attribute name="F_FONT_SIZE" type="xsd:short" />
        <xsd:attribute name="F_FONT_STRIKETHROUGH" type="xsd:boolean"
/>
        <xsd:attribute name="F_FONT_UNDERLINE" type="xsd:boolean" />
        <xsd:attribute name="F_FORECOLOR" type="xsd:long" />
        <xsd:attribute name="F_HASBORDER" type="xsd:boolean" />
        <xsd:attribute name="F_HEIGHT" type="xsd:double" />
        <xsd:attribute name="F_ID" type="xsd:long" use="required" />
```

```
            <xsd:attribute name="F_LEFT" type="xsd:double" />

            <xsd:attribute name="F_LINE_BACKMODE" type="xsd:short" />

            <xsd:attribute name="F_LINE_COLOR" type="xsd:long" />

            <xsd:attribute name="F_LINE_END_X" type="xsd:double" />

            <xsd:attribute name="F_LINE_END_Y" type="xsd:double" />

            <xsd:attribute name="F_LINE_START_X" type="xsd:double" />

            <xsd:attribute name="F_LINE_START_Y" type="xsd:double" />

            <xsd:attribute name="F_LINE_STYLE" type="xsd:short" />

            <xsd:attribute name="F_LINE_WIDTH" type="xsd:short" />

            <xsd:attribute name="F_MODIFYDATE" type="xsd:dateTime" />

            <xsd:attribute name="F_MULTIPAGETIFFPAGENUMBER"
type="xsd:short" />

            <xsd:attribute name="F_NAME" type="xsd:string" use="required"
/>

            <xsd:attribute name="F_ORDINAL" type="xsd:long" />

            <xsd:attribute name="F_PAGENUMBER" type="xsd:short"
use="required" />

            <xsd:attribute name="F_ROTATION" type="xsd:short" />

            <xsd:attribute name="F_TEXT_BACKMODE" type="xsd:short" />

            <xsd:attribute name="F_TOP" type="xsd:double" />

            <xsd:attribute name="F_WIDTH" type="xsd:double" />

            <xsd:attribute name="F_SUBCLASS" type="xsd:string" />

            <xsd:attribute name="F_SCALE" type="xsd:double" />

            <xsd:attribute name="F_VIEWOPTION" type="xsd:short" />

        </xsd:complexType>

        <!--The following complex types define the annotation security.

           NOTE:  The security schema is removed from this xsd; it is now
imported from FnSecurity.xsd.

           Use the following command to generate the vb file for the
classes:

           xsd /classes /language:vb fndocannolist.xsd FnSecurity.xsd

           then remove the classes for security from this vb file to avoid
compilation errors since they have been defined in FnSecurity.vb . -->

        <!--The permission collection type. -->

        <!--The Permission item.-->
```

```
    <!--Enumeration for IS access rights.-->

    <!-- Enumeration for DS access rights.-->

    <!--Enumeration for both CS and IS access rights.

       This is tried to resolve the problem of using the same tag for
both IS and CS security.-->

    <!--Enumeration for the security type.-->

    <!--Enumeration for the client permission right to change the
security values:

       None   client has not right to change the security entries.

       Change client has the right to change the permission entries.

       Admin  same as Change plus: (To be added)-->

    <!--Enumeration for the system type.-->

    <xsd:complexType name="FnAnnoDefPermissionType">

        <xsd:sequence>

            <xsd:element name="security" type="sec:securitytype" />

        </xsd:sequence>

    </xsd:complexType>

</xsd:schema>
```

FnSecurity.xsd

```
<xsd:schema id="FnSecurity"
targetNamespace="http://tempuri.org/FnSecurity.xsd"
elementFormDefault="qualified"
xmlns="http://tempuri.org/FnSecurity.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

       <xsd:element name="security" type="securitytype"></xsd:element>

       <xsd:complexType name="securitytype">

              <xsd:sequence>

                     <xsd:element name="securityobject"
type="securityobjecttype" />

              </xsd:sequence>

       </xsd:complexType>

       <xsd:complexType name="securityobjecttype">

              <xsd:sequence>

                     <xsd:element name="permission"
type="permissiontype" minOccurs="0" maxOccurs="unbounded" />

              </xsd:sequence>
```

```
                <xsd:attribute name="libraryid" type="xsd:string" />
                <xsd:attribute name="systemtype" type="FnSystemType" />
                <xsd:attribute name="objectid" type="xsd:string" />
                <xsd:attribute name="objecttype" type="xsd:string" />
                <xsd:attribute name="clientpermission"
type="FnClientPermissionType" />
        </xsd:complexType>
        <xsd:complexType name="permissiontype">
                <xsd:sequence />
                <xsd:attribute name="id" type="xsd:string" />
                <xsd:attribute name="name" type="xsd:string" />
                <xsd:attribute name="type" type="FnPermissionType" />
                <xsd:attribute name="level" type="FnAccessLevel" />
                <xsd:attribute name="updatetype" type="FnUpdateType" />
        </xsd:complexType>
        <xsd:simpleType name="ISAccessLevel">
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="read" />
                        <xsd:enumeration value="write" />
                        <xsd:enumeration value="append" />
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="CSAccessLevel">
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="none" />
                        <xsd:enumeration value="viewer" />
                        <xsd:enumeration value="author" />
                        <xsd:enumeration value="owner" />
                        <xsd:enumeration value="admin" />
                </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType name="FnAccessLevel">
                <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="none" />
                        <xsd:enumeration value="viewer" />
                        <xsd:enumeration value="author" />
                        <xsd:enumeration value="owner" />
                        <xsd:enumeration value="admin" />
                        <xsd:enumeration value="read" />
                        <xsd:enumeration value="write" />
```

```
                        <xsd:enumeration value="append" />
                    </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType name="FnPermissionType">
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="user" />
                        <xsd:enumeration value="group" />
                    </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType name="FnClientPermissionType">
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="none" />
                        <xsd:enumeration value="change" />
                        <xsd:enumeration value="admin" />
                    </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType name="FnSystemType">
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="idmis" />
                        <xsd:enumeration value="idmds" />
                    </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType name="FnUpdateType">
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="none" />
                        <xsd:enumeration value="change" />
                        <xsd:enumeration value="add" />
                        <xsd:enumeration value="remove" />
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:schema>
```

Appendix C

FileNet End User Software License Notice

FileNet Notice to End User - A Software License is Required Prior to Use.

BEFORE COMPLETING INSTALLATION OR USING THIS SOFTWARE, CAREFULLY READ THIS NOTICE.

THIS SOFTWARE IS THE PROPRIETARY INTELLECTUAL PROPERTY OF FILENET CORPORATION (OR ITS SOFTWARE SUPPLIERS) AND USE OF ANY PORTION OF THE SOFTWARE IS ONLY PERMITTED IF YOU HAVE A VALID WRITTEN LICENSE AGREEMENT WITH FILENET.

You may have a valid FileNet software license agreement if:

1. Your EMPLOYER and FileNet have entered into a written license agreement; or

2. Your EMPLOYER and an authorized FileNet partner have entered into a written license agreement.

If you do not have a valid license agreement to use the software, then terminate the installation of this software, promptly delete any FileNet software files from your computer, and return the software media and all other related items to: FileNet Corporation, 3565 Harbor Blvd., Costa Mesa, CA 92626-1420, USA.

FileNet only licenses use of Software to end user. Nothing in this Notice or any written license agreement constitutes a sale of the software. FileNet reserves all rights.

Notice to U.S. Government of Restricted Rights.

The software and documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101 (October 1995), consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 (September 1995) or 48 C.F.R. §227.7202-1 through 7202–4(June 1995) as applicable.

Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items, and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions of the applicable software license agreement.

Contractor/manufacturer is: FileNet Corporation, 3565 Harbor Blvd., Costa Mesa, California 92626-1420, USA.

Index