*e*Index Global Identifier Product Suite*

Java Programmer's Guide for e*Index™ Active Integration

Release 5.0.5 for Schema Run-time Environment (SRE)

SEEBEYOND

Version 20050126115600

Table of Contents

Chapter 1

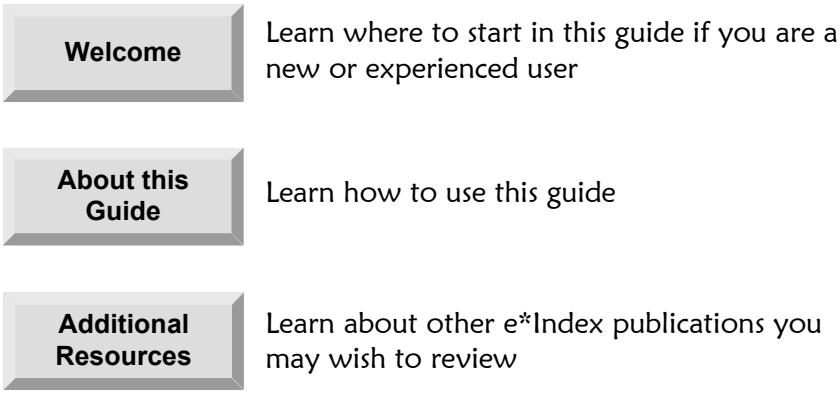
Introduction

About this Chapter

Overview

This Introduction welcomes new and experienced e*Index™ Global Identifier (e*Index) users and explains the structure of this guide and how to use this guide.

The following diagram illustrates each major topic in this chapter.



Welcome

About e*Index

e*Index is SeeBeyond's enterprise-wide master person index, designed to help you maintain information about your members, and to ensure that the information is the most current and accurate data available. e*Index works together with SeeBeyond's e*Gate and the Database e*Ways to transfer information among various computer systems within your business. Using the Java API functions provided with e*Index, you can create your own Java programs to insert and update data in the e*Index database. The Java functions ensure that data integrity is maintained and that records are identified and cross-referenced accurately.

To New Users

If you are new to e*Index, you should browse through this guide before you begin to use the Java APIs provided with e*Index. Chapters 2 and 3 of this guide provide background and explanatory information you may need to understand before beginning to work with the e*Index Java APIs. After reading this overview information and the conceptual information, you will be ready to create your own customized programs using the standard set of e*Index Java APIs described in chapters 4 and 5.

To Established Users

If you are a more advanced e*Index user, you may prefer to use this guide as a quick reference to find information about forgotten or unfamiliar Java functions. If you know what you need to do, but do not remember exactly how to do it, you can easily find what you need in the Table of Contents. Or, you can browse through the guide and find the appropriate background information or API description by scanning headings and titles.

Recommended Reading

This guide does not explain standard message processing for e*Index, although it is very helpful to understand e*Index processing concepts. Before working with the Java APIs for e*Index Active Integration, SeeBeyond recommends that you read Chapter 2, "Understanding Operational Processes" in the *e*Index Global Identifier Technical Reference*. This chapter provides general information about the architecture and components of e*Index, data processing flow, and e*Index database tables. This information can be very useful when designing the active applications to use with e*Index.

About this Guide

Purpose

This guide provides the information you need to create programs in Java that use the matching logic of e*Index to insert and update information in the e*Index database. It also provides an overview of the data processing flow for e*Index, and describes each e*Index Java class and method.

Scope

This guide includes:

- Conceptual information about the Java APIs for e*Index Active Integration
- A complete reference to the Java APIs for e*Index Active Integration, including descriptions, syntax, parameters, return values, and examples

This guide does not explain how to perform any of the tasks listed below. For a list of publications that contain this information, see "Additional Resources" at the end of this chapter.

- How to use the GUI front-end for e*Index applications
- How to install and configure e*Index
- How to configure the e*Index Schema
- How to create and implement Java programs

Intended Audience

This guide should be read by any Java programmers who need to include the Java APIs for e*Index Active Integration in their Java programs. To understand the information in this guide, a reasonably good understanding of the following areas is recommended:

- Java programming concepts
- Database connectivity using Java
- Your e*Index environment
- e*Index data processing concepts
- The data formats used by the systems you work with

Using this Guide

Before you begin to use this guide:

1. You may want to review information presented in other e*Index guides. See "Additional Resources" at the end of this chapter for a list of available publications.
2. Skim through this guide to familiarize yourself with the locations of essential functions you need to use or API descriptions you need to understand. Each chapter begins with a simple graphic that identifies the information contained in the chapter.

Document Organization

This guide is divided into five chapters that cover the topics shown below.

Chapter	Topics
Chapter 1, Introduction	■ Welcome ■ About this Guide ■ Additional Resources
Chapter 2, Overview of the Java API	■ Learning About e*Index Active Integration ■ Learning About the Java APIs for e*Index Active Integration ■ Learning About Active Integration Java Classes
Chapter 3, Implementing e*Index Active Integration	■ Working with Supporting Files ■ About Active Integration Database Tables ■ Working with the Java API
Chapter 4, Person Package	■ Learning About the Person Package ■ Class and method descriptions
Chapter 5, Exception Package	■ Learning About the Exception Package ■ Class and method descriptions

Conventions

Before you read this guide, it's important to understand the typographic, icon, special notation, and other conventions used in this guide.

Typographic Conventions

The following typographic conventions are used in this and other e*Index publications.

Item	Convention	Example
Book titles	Title caps, italic	See the e*Index *Global Identifier User Guide*
Chapter titles (and section titles within chapters)	Title caps, in quotation marks	See Chapter 3, "Implementing e*Index Active Integration" See "Working with the Sample Code" later in this chapter
New terms	Italic	This class reads a *property file*.
Typed command syntax	Bold for constants Bold-italic for user-specified values Brackets denote optional values	Type **mkdir *release***
Java method syntax	Shaded Italic for parameters Brackets denote optional values	`public boolean equals(java.lang.Object obj)`

Icon and Special Notation Conventions

The following conventions are used in this and other e*Index publications to identify special types of information.

Icon or Notation	Type of information
Note	Supplemental information that is helpful to know, but not essential to completing a particular task.
Tip	Information that helps you to apply techniques and procedures described in the text to your specific needs. May also suggest alternative methods.
Important!	Information that is essential to the completion of a task.
Caution!	Advises you to take specific action to avoid loss of data.
	Indicates the beginning of a step-by-step instruction.
	Specifies a task to perform before you begin a step-by-step instruction.
	Indicates a cross-reference to other sections of the guide or to other publications.

Additional Resources

SeeBeyond has developed a suite of e*Index user guides and related publications that are distributed in an electronic library.

- *e*Index Global Identifier User's Guide*
  Helps e*Index quality workstation users to perform database maintenance tasks, such as merging and unmerging records, finding and resolving potential duplicates, adding and updating records, and viewing the audit trail.
- *e*Index Administrator User's Guide*
  Helps system administrators configure system parameters, customize e*Index, work with Vality rule set files, and processing codes. This guide also describes how to maintain the information in the database that is used to populate the drop-down lists in the e*Index.
- *e*Index Security User's Guide*
  Helps system administrators add users and user groups to e*Index, to grant security permissions to users and user groups, to maintain user and user group information, and to configure certain system parameters.
- *e*Index Global Identifier Technical Reference*
  Describes message processing for e*Index, as well as database tables and e*Index Monk APIs. This guide also provides a complete listing of e*Index Monk APIs and functions, along with a description, parameters, syntax, return values, and examples for each.
- *e*Index Initial Load User's Guide*
  Provides the background information and instructions that system and database administrators need in order to load legacy data into the e*Index database, including a description of the expected data format and the schema files included with the load program.
- *Working with Reports for e*Index Global Identifier*
  Provides background information about the GUI and standard reports provided with e*Index, and explains how to modify and run the standard reports (for Oracle installations only).
- *e*Index Global Identifier Installation Guide*
  Helps system and database administrators install a new e*Index environment for the current release, including e*Index Schema files, the e*Index GUI, and database installation.
- *e*Index Global Identifier Upgrade Guide*
  Helps system and database administrators upgrade an existing e*Index environment to the most current release from version 4.1.2 or later, including e*Index Schema files, the e*Index GUI, and database upgrades.

Chapter 2

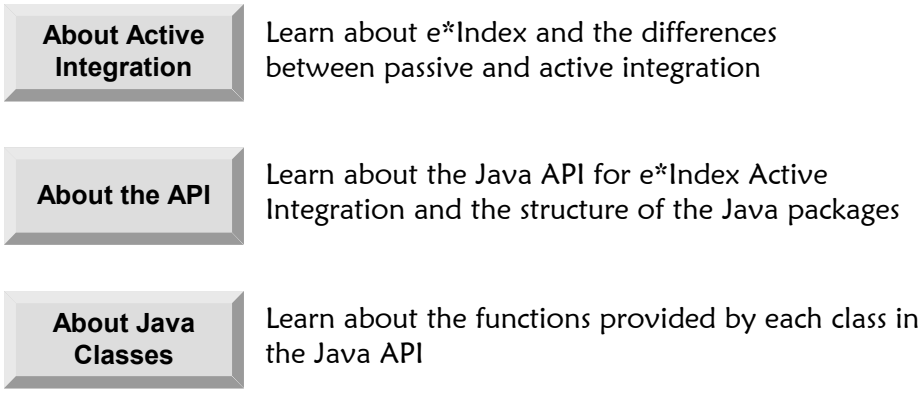
Overview of the Java API

About this Chapter

Overview

This chapter presents an overview of the classes, methods, and fields defined in the Java APIs for e*Index Active Integration, along with information about passive and active implementations of e*Index.

The following diagram illustrates each major topic in this chapter.



Learning About e*Index Active Integration

Overview

This section of the chapter provides background information about e*Index and active integration.

About e*Index

e*Index is an enterprise-wide index that maintains data and enables accurate identification of its members as they participate throughout a business enterprise. e*Index centralizes the identifying and demographic information for all members in one shared index, so that all individual look-ups and data retrievals obtain the most recent information on each person. e*Index uses a single data source regardless of the location or computer system from which member information is received. e*Index is able to cross-reference a member's records throughout several systems by assigning each member a unique global identifier.

e*Index was designed specifically to support geographically dispersed sites and disparate information systems across an enterprise, as well as various applications from multiple vendors. Maintaining a centralized database for multiple systems enables e*Index to integrate data within the enterprise while allowing local systems to continue operating independently.

Passive versus Active Integration

You can implement e*Index in two different modes: passive and active. In a passive implementation, e*Index does not directly interact with the systems that provide the data for the e*Index database, such as a registration system. All of the matching, identification, and cross-referencing are performed behind the scenes, after a record has already been added to or updated in a local system. Legacy system users do not interact directly with the e*Index system. Administrators use the Quality Workstation to monitor data; view assumed matches, potential duplicates, and audit trails; and to resolve potential duplicate records.

In an active implementation, you can use the Java APIs to create a program that provides access to both customized web-based applications and the e*Index system. The custom applications can be designed to allow a user to search for person data, update existing person records, and insert new person records directly in the e*Index database. This reduces the possibility of creating duplicate records that will later need to be analyzed and resolved. In active mode, administrators still use the Quality Workstation to monitor and maintain data.

Learning About the Java APIs for e*Index Active Integration

Overview

SeeBeyond created the Java APIs for e*Index Active Integration in order to enable highly customizable active implementations of e*Index. The APIs are provided in Java to allow you more flexibility when designing your active applications for e*Index. The APIs include a set of functions that allow you to create web-based applications based on a subset of e*Index functions. Using the active integration APIs, these web-based applications can interact with the e*Index database by searching for existing person records, viewing person information, adding new person records, or updating existing person records. The active integration APIs are multi-threaded, allowing multiple users to access the e*Index system through web-based browsers.

About the Active Integration Java Packages

The Java APIs for e*Index Active Integration are provided in two separate Java packages, each containing methods you can combine to perform actions of a specific type against the e*Index database. Each package, along with its associated classes, methods, and fields, is described in its own chapter. The packages are:

- **Person Package**
  The Person Package (**com.stc.eIndex.active.person**) provides the functions needed to lookup and view person information, to insert new person records into the e*Index database, and to update person records that already exist in the e*Index database. It also provides functions to read system parameter and database information, handle connections to the database, define enumerations, verify security information, look up country-specific attributes, look up display configurations (Oracle databases only), and generate local ID numbers (Oracle databases only).
- **Exception Package**
  The Exception Package (**com.stc.eIndex.active.exception**) provides the functions needed for exception handling during data processing. The methods in this package are used to throw exceptions when specific errors occur and to provide detailed messages about the errors.

Active Integration Java Class Hierarchy

The Java APIs for e*Index Active Integration are extensions of the Java class **java.lang.Object**. The outline below illustrates the hierarchy of classes provided with the Active Integration APIs. Not all of these classes are available for a SQL Server e*Index database implementation (for a list of classes not available on SQL Server, see "A Note About SQL Server Implementations" later in this chapter).

- class java.lang.Object
  - class com.stc.eIndex.active.person.ActiveField
  - class com.stc.eIndex.active.person.ActiveFieldFill
  - class com.stc.eIndex.active.person.ActiveFunction
  - class com.stc.eIndex.active.person.ActiveLookup
  - class com.stc.eIndex.active.person.ActiveSet
  - class com.stc.eIndex.active.person.CityState
  - class com.stc.eIndex.active.person.CodeDisplay
  - class com.stc.eIndex.active.person.CodeLookup
  - class com.stc.eIndex.active.person.ControlKey
  - class com.stc.eIndex.active.person.CountryOptionLookup
  - class com.stc.eIndex.active.core.DataObject
    - class com.stc.eIndex.active.person.CountryOption
    - class com.stc.eIndex.active.person.DemographicsRO
      - class com.stc.eIndex.active.person.Demographics
        - class com.stc.eIndex.active.person.Person
        - class com.stc.eIndex.active.person.SearchParameters
    - class com.stc.eIndex.active.core.DynamicDataObject
      - class com.stc.eIndex.active.person.DemographicsResultRO (implements java.lang.Comparable)
    - class com.stc.eIndex.active.person.EiSystem
    - class com.stc.eIndex.active.person.PersonDependent
      - class com.stc.eIndex.active.person.Address
      - class com.stc.eIndex.active.person.Alias
      - class com.stc.eIndex.active.person.Audit
      - class com.stc.eIndex.active.person.AuxId
      - class com.stc.eIndex.active.person.LocalId
      - class com.stc.eIndex.active.person.Phone
    - class com.stc.eIndex.active.person.Transaction
  - class com.stc.eIndex.active.person.EiBOFactory
  - class com.stc.eIndex.active.person.EiConnection (implements java.sql.Connection)
  - class com.stc.eIndex.active.person.EiServer
  - class com.stc.eIndex.active.person.EiSystemBO
  - class com.stc.eIndex.active.person.EnumCodeType

- class com.stc.eIndex.active.person.EnumCountryOption
- class com.stc.eIndex.active.person.EnumLocalIdStatus
- class com.stc.eIndex.active.person.EnumPersonStatus
- class com.stc.eIndex.active.person.LocalIdBO
- class com.stc.eIndex.active.person.LocalIdGenerator
- class com.stc.eIndex.active.person.PersonBO
- class com.stc.eIndex.active.person.PersonSearchResult
- class com.stc.eIndex.active.person.PredefinedMsg
- class com.stc.eIndex.active.person.PredefinedMsgRegistry
- class com.stc.eIndex.active.person.Security
- class com.stc.eIndex.active.person.Ssn
- class java.lang.Throwable (implements java.io.Serializable)
  - class java.lang.Exception
    - class com.stc.eIndex.active.exception.EiException
      - class com.stc.eIndex.active.exception.EiException.DataNotFound
      - class com.stc.eIndex.active.exception.EiException.DeleteRow
      - class com.stc.eIndex.active.exception.EiException.EmptyString
      - class com.stc.eIndex.active.exception.EiException.ExceedMaxStringLength
      - class com.stc.eIndex.active.exception.EiException.GeneralException
      - class com.stc.eIndex.active.exception.EiException.InvalidData
      - class com.stc.eIndex.active.exception.EiException.InvalidHistory
      - class com.stc.eIndex.active.exception.EiException.InvalidLocalIdStatus
      - class com.stc.eIndex.active.exception.EiException.InvalidParameter
      - class com.stc.eIndex.active.exception.EiException.InvalidPersonStatus
      - class com.stc.eIndex.active.exception.EiException.InvalidTimestamp
      - class com.stc.eIndex.active.exception.EiException.InvalidUid
      - class com.stc.eIndex.active.exception.EiException.InvalidUniqueKeyQuery
      - class com.stc.eIndex.active.exception.EiException.MatchException
      - class com.stc.eIndex.active.exception.EiException.NonUpdateableFieldModified
      - class com.stc.eIndex.active.exception.EiException.RequiredFieldIsNull
      - class com.stc.eIndex.active.exception.EiException.UniqueKeyException
      - class com.stc.eIndex.active.exception.EiException.UpdateRow
    - class java.lang.RuntimeException
      - class com.stc.eIndex.active.exception.EiRuntimeException
- class com.stc.eIndex.active.person.Transaction
- class com.stc.eIndex.active.person.Uid (implements java.io.Serializable)
- class com.stc.eIndex.active.person.ZipCodeLookup

Interface Hierarchy

- interface com.stc.eIndex.active.person.EiNull

A Note About SQL Server Implementations

Some of the classes in the hierarchical list above are not available to SQL Server implementations of e*Index. This includes all of the classes used to define and retrieve information about active fields, functions, and sets, as well as the class used to generate local ID numbers. The following classes are not available in a SQL Server implementation.

- class com.stc.eIndex.active.person.ActiveField
- class com.stc.eIndex.active.person.ActiveFieldFill
- class com.stc.eIndex.active.person.ActiveFunction
- class com.stc.eIndex.active.person.ActiveLookup
- class com.stc.eIndex.active.person.ActiveSet
- class com.stc.eIndex.active.person.LocalIdGenerator

Java Components at a Glance

The table below lists all the Java packages provided with e*Index Active Integration, along with their associated classes, methods, and fields.

Table 2- 1: Active Integration Person Package

This class …	includes these methods …	and these fields …
ActiveField	getActiveFieldFillEnumeration getDbColumnName getDefaultValue getFieldLabel getFieldName getFuncCode getRequired getSetCode getTableName	
ActiveFieldFill	getFieldName getFillTrigger getFillValue getFunctionCode	
ActiveFunction	getFunctionCode getFunctionDescription getSetCode	

This class …	includes these methods …	and these fields …
ActiveLookup	getActiveFields getActiveFieldsByUser getActiveFunctions getActiveSet getInstance	
ActiveSet	GetAutoAssign getSetCode getSetDescription getSystemCode	
Address	Address getAddress1 getAddress2 getAddress3 getAddress4 getAddressType getCity getCountry getCounty getHouseNumber getPostalCode getPostalCodeExt getStateOrProvince getStreetDir getStreetName getStreetType setAddress1 setAddress2 setAddress3 setAddress4 setCity setCountry setCounty setPostalCode setPostalCodeExt setStateOrProvince setStreetDir toString	
Alias	Alias equals getFirstName getLastName getMiddleName getUid	

This class …	includes these methods …	and these fields …
	setFirstName setLastName setMiddleName toString	
Audit	Audit getDetail getFunct getTimestamp getUid1 getUid2 getUserId setDetail setFunct setTimestamp setUid2 setUserId	
AuxId	AuxId equals getAuxIdType getId setAuxIdType setId toString	
CityState	getCity getCounty getResidenceCode getState	
CodeDisplay	getCode getCommonDetailId getCommonHeaderId getCreateDate getCreateUserId getDisplay getReadOnly	
CodeLookup	getCodeDisplayPairs getDisplayValue getInstance	
ControlKey	getCheckSumLength getCountry getDateFormat getDobSearchRange getDuplicateSearchLimit getDuplicateThreshold	

This class …	includes these methods …	and these fields …
	getEndTime getExactSearchLength getInstance getMatchThreshold getMaximumThreshold getMinimumThreshold getStartTime getUidLength isAllowNumericEnabled isAssumeMatchEnabled isAuditOnOff isBlankOnUpdateEnabled isDobYearRequired isDuplicateCheckEnabled isExtensiveSearchEnabled isMixedCaseEnabled isOneExactMatchEnabled isSameFacilityReportEnabled isSeeDeactivatedEnabled isSeeMergedEnabled isShortIdEnabled	
Demographics	Demographics clearAll copyDemographics setCitizenship setClass1 setClass2 setClass3 setClass4 setClass5 setDate1 setDate2 setDate3 setDate4 setDate5 setDateOfDeath setDeath setDeathCertificate setDistrictOfResidence setDob setDriversLicenseNumber setDriversLicenseState setEthnic	

This class …	includes these methods …	and these fields …
	setFatherName setFirstName setGender setLanguage setLastName setLgaCode setMaidenName setMaritalStatus setMiddleName setMilitaryBranch setMilitaryRank setMilitaryStatus setMotherMaidenName setMotherName setNationality setPensionExpirationDate setPensionNumber setPersonCategoryCode setPobCity setPobCountry setPobState setRace setReligion setRepatriationNumber setSpouseName SetSsn setString1 setString10 setString2 setString3 setString4 setString5 setString6 setString7 setString8 setString9 setSuffix setTitle setVeteranStatus setVipFlag	

This class …	includes these methods …	and these fields …
DemographicsResultRO	compareTo getCitizenship getClass1 getClass2 getClass3 getClass4 getClass5 getComparisonScore getDate1 getDate2 getDate3 getDate4 getDate5 getDateOfDeath getDeath getDeathCertificate getDistrictOfResidence getDob getDriversLicenseNumber getDriversLicenseState getEthnic getFatherName getFirstName getGender getLanguage getLastName getLgaCode getMaidenName getMaritalStatus getMiddleName getMilitaryBranch getMilitaryRank getMilitaryStatus getMotherMaidenName getMotherName getNationality getPensionExpirationDate getPensionNumber getPobCity getPobCountry getPobState getRace getReligion getRepatriationNumber	

This class …	includes these methods …	and these fields …
	getSpouseName getSsn getStatus getString1 getString10 getString2 getString3 getString4 getString5 getString6 getString7 getString8 getString9 getSuffix getTitle getUid getVeteranStatus getVipFlag toString	
DemographicsRO	getCitizenship getClass1 getClass2 getClass3 getClass4 getClass5 getDate1 getDate2 getDate3 getDate4 getDate5 getDateOfDeath getDeath getDeathCertificate getDistrictOfResidence getDob getDriversLicenseNumber getDriversLicenseState getEthnic getFatherName getFirstName getGender getLanguage	

This class …	includes these methods …	and these fields …
	getLastName getLgaCode getMaidenName getMaritalStatus getMiddleName getMilitaryBranch getMilitaryRank getMilitaryStatus getMotherMaidenName getMotherName getNationality getPensionExpirationDate getPensionNumber getPersonCategoryCode getPobCity getPobCountry getPobState getRace getReligion getRepatriationNumber getSpouseName getSsn getStatus getString1 getString10 getString2 getString3 getString4 getString5 getString6 getString7 getString8 getString9 getSuffix getTitle getVeteranStatus getVipFlag	
EiBOFactory	getEiSystemBOInstance getLocalIdBOInstance getPersonBOInstance	
EiConnection	clearWarnings	

This class …	includes these methods …	and these fields …
	close commit createStatement getAutoCommit getCatalog getEiServer getMetaData getStartTransactionStatement getTransactionIsolation getTypeMap getWarnings isClosed isReadOnly nativeSQL prepareCall prepareStatement rollback setAutoCommit setCatalog setReadOnly setTransactionIsolation setTypeMap	
EiNull (interface)		BOOLEAN FLOAT LONG SQL_DATE STRING TIMESTAMP
EiServer	EiServer getConnection getDbTimestamp getEiBOFactory getLogicalName getProperties hashCode lookupEiServer	
EiSystem	getAddress1 getAddress2 getAllowLessIdLength getCity getCountry getCounty getCreateDate	

This class …	includes these methods …	and these fields …
	getDescription getIdLength getLocalIdFormat getNextLocalId getRegionCode getState getStatus getSystemCode getZip getZipExt isFormatMatch isLocalIdMaskEnabled isSystemMaskEnabled	
EiSystemBO	getSystem getSystemDescription getSystemEnumeration	
EnumCodeType		ADDRESS_TYPE CITIZENSHIP COUNTRY DEPT DISTRICT_OF_RESIDENCE DRIVER_LICENSE_ISSUER ETHNICITY EVENT EVENT_NOTIFICATION GENDER LANGUAGE MARITAL_STATUS NATIONALITY PERSON_CATEGORY PERSON_STATUS PHONE_TYPE RACE REGION RELIGION STATE SUFFIX SYSTEM TITLE VETERAN_STATUS VIP
EnumCountryType		ADDRESS_SEARCH COLUMN_FORMAT

This class …	includes these methods …	and these fields …
		GROUP_LABEL SUMMARY_TAB TAB_LABEL
EnumLocalIdStatus	getCode getEnumeration toString	ACTIVE DEACTIVATED MERGED
EnumPersonStatus	getEnumeration toString	ACTIVE DEACTIVATED MERGED
LocalID	LocalId equals getLocalId getStatus getSystemCode getUid	
LocalIdBO	getLocalIdEnumeration isActive isDeactivated isMerged lookupLocalId	
LocalIdGenerator	getInstance getNextLocalId	
Person	Person clearAll getAddressEnumeration getAliasEnumeration getAuxIdEnumeration getDemographics getLocalIdEnumeration getPhoneEnumeration getUid setDemographics	
PersonBO	addAddress addAlias addAudit addAuxId addLocalId addPerson addPhone addPredefinedComment addUserDefinedComment getActiveUid getDemographics	

This class …	includes these methods …	and these fields …
	hasAddress hasAlias hasAuxId hasLocalId hasPhone isActivePerson isDeactivatedPerson isMergedPerson loadPerson lookupUid processPerson searchAlpha searchGeneral searchPhonetic uidLookupByAuxId updatePerson	
PersonSearchResult	first getResultRow last next previous setResultRow size	
Phone	equals getPhoneExtension getPhoneNumber getPhoneType getUid setPhoneExtension setPhoneNumber toString	
PredefinedMsg	getCode getDescription	
PredefinedMsgRegistry	getInstance getPredefinedMsgEnumeration	
SearchParameters	SearchParameters setUid	
Security	getInstance isUserValid	
Ssn	Ssn equals toString	NULL
Transaction	Transaction	

This class …	includes these methods …	and these fields …
	getDepartment getFunction getSource getSystem getTerminalId getTimeStamp getUserId toString	
Uid	Uid equals toString	NULL
ZipCode	getCityState getInstance	

Table 2- 2: Active Integration Exception Package

This class …	includes these methods …
EiException	None
EiException.DataNotFound	EiException.DataNotFound
EiException.DeleteRow	EiException.DeleteRow
EiException.EmptyString	EiException.EmptyString
EiException.ExceedMaxStringLength	EiException.ExceedMaxStringLength
EiException.GeneralException	EiException.GeneralException
EiException.InvalidData	EiException.InvalidData
EiException.InvalidHistory	EiException.InvalidHistory
EiException.InvalidLocalIdStatus	EiException.InvalidLocalIdStatus
EiException.InvalidParameter	EiException.InvalidParameter
EiException.InvalidTimestamp	EiException.InvalidTimestamp
EiException.InvalidUid	EiException.InvalidUid
EiException.InvalidUniqueKeyQuery	EiException.InvalidUniqueKeyQuery
EiException.MatchException	EiException.MatchException
EiException.NonUpdateableFieldModified	EiException.NonUpdateableFieldModified
EiException.RequiredFieldIsNull	EiException.RequiredFieldIsNull
EiException.UniqueKeyException	EiException.UniqueKeyException
EiException.UpdateRow	EiException.UpdateRow
EiRuntimeException	EiRuntimeException

Learning About Active Integration Java Classes

Overview

This section of the chapter provides background information about the classes provided for the Java APIs for e*Index Active Integration. For detailed information about a specific class, refer to the appropriate class section in chapter 4 or 5 of this guide.

About Active Integration Java Classes

The Java APIs for e*Index Active Integration include several classes that provide the necessary e*Index functions for your custom programs. Most of the classes are used to search for person records, insert new person records, or update existing person records. Classes are also included to retrieve instances, read system parameters, provide exception handling, provide connectivity logic, and so on. The purpose of each class is described in Table 2-3 on page 2-19 and Table 2-4 on page 2-25.

Some of these classes are only available in Oracle implementations. The classes not available to SQL Server implementations include:

- ActiveField
- ActiveFieldFill
- ActiveFunction
- ActiveLookup
- ActiveSet
- LocalIdGenerator

Available Active Integration Java Classes

Several Java classes are defined for e*Index Active Integration, including classes representing demographic records, exceptions, code table values, enumerated lists, and so on. If you know the name of a Java class, you can use this chart to identify the purpose of the class.

Table 2-3: Active Integration Java Classes

Use Methods in this Class …	to perform this action …
ActiveField	Retrieve information about the fields defined for business rules scenarios. This class is only available to Oracle database implementations.

Use Methods in this Class …	to perform this action …
ActiveFieldFill	Retrieve an enumeration of field fill and field trigger definitions for the ActiveField class. This class is only available to Oracle database implementations.
ActiveFunction	Retrieve information about the functions defined for business rules scenarios. This class is only available to Oracle database implementations.
ActiveLookup	Look up the sets, functions, and field information defined for business rules scenarios. This class is only available to Oracle database implementations.
ActiveSet	Retrieve information about the settings defined for business rules scenarios. This class is only available to Oracle database implementations.
Address	Create a new Address object and retrieve individual fields from an Address object.
Alias	Create a new Alias object and retrieve individual fields from an Alias object.
Audit	Create a new Audit object.
AuxId	Create a new AuxId object and retrieve individual fields from an AuxId object.
CityState	Retrieve a city and state pair using the zip code lookup function. You can also retrieve the county and residence code for the specified zip code.
CodeDisplay	Retrieve a processing code and display value pair using the code lookup function.
CodeLookup	Find the display value for a specified processing code.
ControlKey	Retrieve the values of the e*Index control keys (found in the database table *ui_control*).
CountryOption	Retrieve the values of specific fields contained in an option set returned by CountryOptionLookup.
CountryOptionLookup	Look up country-specific attribute definitions by control type or by control type, country, and option name.
Demographics	Retrieve or modify a record's demographic information, or insert new demographic data.

Use Methods in this Class …	to perform this action …
DemographicsResultRO	Retrieve and display a partial set of demographic data for a specific record. This represents the result set returned from a demographic search and the properties cannot be modified.
DemographicsRO	Retrieve a complete set of demographic information for a record. DemographicsRO objects are read-only.
EiBOFactory	Retrieve an instance of a system, local ID. or person business object. This class maintains a singleton pattern for EiSystemBO, LocalIdBO, and PersonBO instances.
EiConnection	Retrieve database connectivity information and modify certain settings. This class wraps standard Java connectivity APIs and provides services for statement caching and tracing.
EiException	Create an e*Index exception with an error message. EiException encloses all of the following classes except EiRuntimeException.
DataNotFound	Create an e*Index exception and error message when the required data cannot be found.
DeleteRow	Create an e*Index exception and error message when a row being deleted cannot be removed. This exception is not currently used.
EmptyString	Create an e*Index exception and error message when an empty string is passed into a field that does not allow empty strings.
ExceedMaxStringLength	Create an e*Index exception and error message when the input value is longer than that allowed in the database column.
GeneralException	Create a general e*Index exception and error message.
InvalidData	Create an e*Index exception and error message when invalid data is passed in.
InvalidHistory	Create an e*Index exception and error message when a history record is invalid. This exception is not currently used.
InvalidLocalIdStatus	Create an e*Index exception and error message when the status of a local ID is not valid.

Use Methods in this Class …	to perform this action …
InvalidParameter	Create an e*Index exception and error message when an invalid parameter is passed into an API.
InvalidPersonStatus	Create an e*Index exception and error message when the status of a person record is not valid.
InvalidTimestamp	Create an e*Index exception and error message when a timestamp is not valid.
InvalidUid	Create an e*Index exception and error message when the specified UID is not valid.
InvalidUniqueKeyQuery	Create an e*Index exception and error message that indicate an error during a query. This exception is not currently used.
MatchException	Create an e*Index exception and error message when an exception is thrown by the Vality matching algorithm.
NonUpdateableField	Create an e*Index exception and error message when an attempt is made to update a read-only field.
RequiredFieldIsNull	Create an e*Index exception and error message when null value is passed into a field that does not allow null values.
UniqueKeyException	Create an e*Index exception and error message when a unique key constraint is violated.
UpdateRow	Create an e*Index exception and error message when there is an error updating a row.
EiNull (interface)	Specify the data type of the field being set to null.
EiRuntimeException	Create an e*Index exception and error message when a runtime exception occurs.
EiServer	Read the properties file. This class is the container for all singleton classes, and must be instantiated before any other active integration APIs are called.
EiSystem	Retrieve information about a specified system from the *ui_facility* table.
EiSystemBO	Create EiSystem objects, retrieve system descriptions, and retrieve an enumeration of all defined systems.

Use Methods in this Class …	to perform this action …
EnumCodeType	Create the enumerations for the CodeLookup class to help look up e*Index processing codes and their display values.
EnumCountryOption	Create the enumerations for the CountryOptionLookup class to help look up country-specific option definitions.
EnumLocalIdStatus	Create the enumerations used by the **getStatus** function in the LocalId class and the **lookupUid** function in the PersonBO class to retrieve the status of a local ID record.
EnumPersonStatus	Create the enumerations used by the **getStatus** method in the DemographicsRO class to retrieve the status of a person record.
LocalId	Create a new LocalId object or display the properties of a LocalId object.
LocalIdBO	Determine the status of the specified local ID and system record.
LocalIdGenerator	Retrieve the next available local identifier for the specified system. The *ui_local_id_generator* table must be populated in order to generate local IDs. This class is only available to Oracle database implementations.
Person	Create a new Person object and display information in a Person object. This class inherits from the Demographics and DemographicsRO classes.
PersonBO	Process person records in the e*Index database, including adding, updating, and searching for person records. This class provides the primary functions for processing person data.
PersonSearchResult	Scroll through the results of an alphanumeric or phonetic search. A PersonSearchResult object is a collection of DemographicsResultRO objects.
Phone	Create a new Phone object and look up information for an existing Phone object.
PredefinedMsg	Retrieve the code and description for predefined messages.
PredefinedMsgRegistry	Retrieve an enumeration of predefined messages from the *ui_canned_msg* table.
SearchParameters	Create an object that defines the criteria for a demographic person search.

Use Methods in this Class …	to perform this action …
Security	Confirm the validity of the specified user login ID and password.
Ssn	Create a new Ssn object representing a person's social security number.
Transaction	Create a new transaction object or retrieve the parameters for a specific transaction, including the transaction type, originating system and source, timestamp, and so on.
Uid	Create a new Uid object representing a unique identification number assigned to a person record by e*Index.
ZipCode	Perform city and state lookups based on the zip code and, optionally, the zip code extension.

Which Class to Use

If you know what you want to do, but you are not sure which e*Index Java API to use, refer to the following chart to look up a task that falls into one of the following categories:

- Connectivity and Initialization
- Instantiation
- Processing Data and Searches
- Enumerations
- Business Rules Scenarios
- Country-specific Options
- Exceptions and Error Messages
- Configuration
- Security

Table 2-4: Active Integration Java Classes

To perform this action …	use Methods in this Class …
Connectivity and Initialization	
Retrieve the values of the e*Index control keys (found in the database table *ui_control*).	**ControlKey**
Retrieve database connectivity information and modify certain settings. This class wraps standard Java connectivity APIs and provides services for statement caching and tracing.	**EiConnection**
Instantiation	
Retrieve an instance of a system, local ID, or person business object. This class maintains a singleton pattern for EiSystemBO, LocalIdBO, and PersonBO instances.	**EiBOFactory**
Read the properties file. This class is the container for all singleton classes, and must be instantiated before any other active integration APIs are called.	**EiServer**
Processing Data and Searches	
Create a new Address object and retrieve individual fields from an Address object.	**Address**
Create a new Alias object, and retrieve individual fields from an Alias object.	**Alias**
Create a new Audit object.	**Audit**
Create a new AuxId object and retrieve individual fields from an AuxId object.	**AuxId**
Retrieve city, state, county, and residence code information using the zip code lookup function.	**CityState**
Retrieve or modify a record's demographic information, or insert new demographic data.	**Demographics**
Retrieve and display a partial set of demographic data for a specific record. This represents the result set returned from a demographic search and the properties cannot be modified.	**DemographicsResultRO**
Retrieve a complete set of demographic information for a record. DemographicsRO objects are read-only.	**DemographicsRO**
Specify the data type of the field being set to null.	***EiNull*** (interface)
Retrieve information about a specified system from the *ui_facility* table.	**EiSystem**

To perform this action …	use Methods in this Class …
Create EiSystem objects, retrieve system descriptions, and retrieve an enumeration of all defined systems.	**EiSystemBO**
Create a new LocalId object or display the properties of a LocalId object.	**LocalId**
Determine the status of the specified local ID and system record.	**LocalIdBO**
Retrieve the next available local identifier for the specified system. The *ui_local_id_generator* table must be populated in order to generate local IDs. This class is only available to Oracle database implementations.	**LocalIdGenerator**
Create a new Person object and display information in a Person object. This class inherits from the Demographics and DemographicsRO classes.	**Person**
Process person records in the e*Index database, including adding, updating, and searching for person records. This class provides the primary functions for processing person data.	**PersonBO**
Scroll through the results of an alphanumeric or phonetic search. A PersonSearchResult object is a collection of DemographicsResultRO objects.	**PersonSearchResult**
Create a new Phone object and look up information for an existing Phone object.	**Phone**
Retrieve the code and description for predefined messages.	**PredefinedMsg**
Retrieve an enumeration of predefined messages from the *ui_canned_msg* table.	**PredefinedMsgRegistry**
Create an object that defines the criteria for a demographic person search.	**SearchParameters**
Create a new Ssn object representing a person's social security number.	**Ssn**
Create a new transaction object or retrieve the parameters for a specific transaction, including the transaction type, originating system and system, timestamp, and so on.	**Transaction**
Create a new Uid object representing a unique identification number assigned to a person record by e*Index.	**Uid**

To perform this action …	use Methods in this Class …
Perform city and state lookups based on the zip code and, optionally, the zip code extension.	**ZipCode**
Enumerations	
Create the enumerations for the CodeLookup class to help look up e*Index processing codes and their display values.	**EnumCodeType**
Create the enumerations used by the **getStatus** function in the LocalId class and the **lookupUid** function in the PersonBO class to retrieve the status of a local ID record.	**EnumLocalIdStatus**
Create the enumerations used by the **getStatus** method in the DemographicsRO class to retrieve the status of a person record.	**EnumPersonStatus**
Business Rules Scenarios	
Retrieve information about the fields defined for business rules scenarios. This class is only available to Oracle database implementations.	**ActiveField**
Retrieve an enumeration of field fill and field trigger definitions for the ActiveField class. This class is only available to Oracle database implementations.	**ActiveFieldFill**
Retrieve information about the functions defined for business rules scenarios. This class is only available to Oracle database implementations.	**ActiveFunction**
Look up the sets, functions, and field information defined for business rules scenarios. This class is only available to Oracle database implementations.	**ActiveLookup**
Retrieve information about the settings defined for business rules scenarios. This class is only available to Oracle database implementations.	**ActiveSet**
Country-specific Options	
Retrieve the values of specific fields contained in an option set returned by CountryOptionLookup.	**CountryOption**
Look up country-specific attribute definitions by control type or by control type, country, and option name.	**CountryOptionLookup**

To perform this action …	use Methods in this Class …
Create the enumerations for the CountryOptionLookup class to help look up country-specific option definitions.	**EnumCountryOption**
Exceptions and Error Messages	
Create an e*Index exception with an error message. EiException encloses all of the following classes except EiRuntimeException.	**EiException**
Create an e*Index exception and error message when the required data cannot be found.	**DataNotFound**
Create an e*Index exception and error message when a row being deleted cannot be removed. This exception is not currently used.	**DeleteRow**
Create an e*Index exception and error message when an empty string is passed into a field that does not allow empty strings.	**EmptyString**
Create an e*Index exception and error message when the input value is longer than that allowed in the database column.	**ExceedMaxStringLength**
Create a general e*Index exception and error message.	**GeneralException**
Create an e*Index exception and error message when invalid data is passed in.	**InvalidData**
Create an e*Index exception and error message when a history record is invalid. This exception is not currently used.	**InvalidHistory**
Create an e*Index exception and error message when the status of a local ID is not valid.	**InvalidLocalIdStatus**
Create an e*Index exception and error message when an invalid parameter is passed into an API.	**InvalidParameter**
Create an e*Index exception and error message when the status of a person record is not valid.	**InvalidPersonStatus**
Create an e*Index exception and error message when a timestamp is not valid.	**InvalidTimestamp**
Create an e*Index exception and error message when the specified UID is not valid.	**InvalidUid**
Create an e*Index exception and error message that indicate an error during a query. This exception is not currently used.	**InvalidUniqueKeyQuery**

To perform this action …	use Methods in this Class …
Create an e*Index exception and error message when an exception is thrown by the Vality matching algorithm.	**MatchException**
Create an e*Index exception and error message when an attempt is made to update a read-only field.	**NonUpdateableField**
Create an e*Index exception and error message when null value is passed into a field that does not allow null values.	**RequiredFieldIsNull**
Create an e*Index exception and error message when a unique key constraint is violated.	**UniqueKeyException**
Create an e*Index exception and error message when there is an error updating a row.	**UpdateRow**
Create an e*Index exception and error message when a runtime exception occurs.	**EiRuntimeException**
Configuration	
Retrieve a processing code and display value pair using the code lookup function.	**CodeDisplay**
Find the display value for a specified processing code.	**CodeLookup**
Provide access to the e*Index control key values (found in the database table **ui_control**).	**ControlKey**
Security	
Confirm the validity of the specified user login ID and password.	**Security**

About Active Integration Java Methods

Several Java methods are defined for e*Index Active Integration, including functions that create demographic lists, search for person records, update person information, display configuration settings, and so on. Many of the methods in the e*Index API are based on standard Java methods, such as the connectivity methods in the EiConnection class and the **toString** and **equals** classes in many of the classes. Several methods are also standard get and set methods. Each method defined for the Java APIs for e*Index Active Integration is described in detail in Chapter 4, "Person Package", and Chapter 5, "Exception Package".

For More Information

Other SeeBeyond publications may help you to learn how to perform tasks associated with creating Java programs for e*Index.

To learn more about …	See …
e*Index architecture and components	"Learning About e*Index" in chapter 2 of the *e*Index Global Identifier Technical Reference*
Standard e*Index processing	"Learning About Event Processing" in chapter 2 of the *e*Index Global Identifier Technical Reference*
The e*Index database	"About the e*Index Database" in chapter 2 of the *e*Index Global Identifier Technical Reference*
Code tables and control keys	The *e*Index Administrator User's Guide*
Standard e*Index functions	The *e*Index Global Identifier User's Guide*

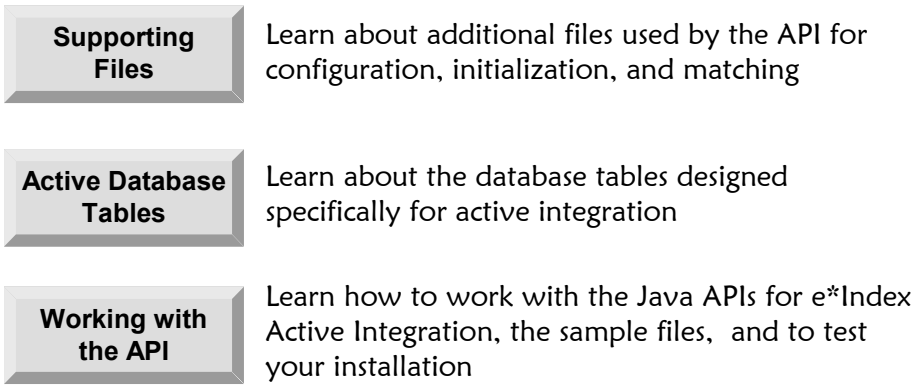
Chapter 3

Implementing e*Index Active Integration

About this Chapter

Overview

This chapter presents the background information that will help you get started creating Java programs using the Java APIs for e*Index Active Integration, including information about the sample files, required files, testing the installation and database connection, and implementing the APIs. The following diagram illustrates each major topic in this chapter.



Working with Supporting Files

Overview

This section of the chapter provides background information about the files that are required to support the Java APIs for e*Index Active Integration. You will need to customize most of these files for your implementation. The supporting files include:

- Server Properties File
- Configuration File
- Vality Rule Set Files

About the Server Properties File

The *server properties file* defines the runtime characteristics of the Java APIs for e*Index Active Integration. This file is read by the EiServer class, which must be instantiated before any other e*Index function requiring the API is called. The properties file may be specified by the EiServer constructor method. For example, if the name of the properties file is **EiServer.properties**, you could instantiate the EiServer class with a line of code similar to the following:

```
EiServer eiServer = new EiServer("EiServer.properties");
```

Alternatively, you can use a Properties object as a parameter to EiServer by first creating a properties object, and then loading the properties information directly into the Properties object. The Properties object is then passed to EiServer as a parameter, as illustrated in the following example:

```
Properties props = new Properties();
... /* loading properties into props */
EiServer eiServer = new EiServer(props)
```

The default properties file for the sample files is named **EiServer.properties**, and you can customize this file for use in your Java application. This file is located in **\<home_dir\sample**. Remember that the directory in which your customized property file is located must be specified in the **CLASSPATH** environment variable on your computer (only if you use the file as a parameter to EiServer). The property file includes the following variables:

- **databaseTimeOut**
  This variable defines the maximum length of time a database statement can be processed before an exception is thrown. Configure this variable to ensure that processes that take a long time do not degrade system performance.
- **databaseMaximumQuerySize**
  This variable defines the maximum number of records that can be retrieved during a weighted person search. Configure this variable to

ensure that searches do not return large result sets, which could slow down performance. Note that the default value, **3**, is set for testing purposes only. The actual value you use should be much larger in order to avoid an exception being thrown.

- **databaseMaximumConnections**
  This variable defines the maximum number of database connections that can be used at one time.
- **databaseMaximumStatementCacheSize**
  This variable defines the maximum number of prepared statements that can be cached per connection.
- **databaseNetworkProtocol**
  This variable defines the network protocol for the connections to the database. For Oracle, this can be set to all protocols supported by Net8. It is only required for the JDBC OCI driver.
- **databaseDriverType**
  This variable defines the Oracle JDBC driver type. The possible values are **thin** and **oci8**. This variable is ignored for SQL Server implementations.
- **databaseUserId**
  This variable defines the user name with which to log in to the e*Index database and through which connections are obtained. The user ID specified must be defined in e*Index security, and must have the appropriate access permissions assigned.
- **databasePassword**
  This variable defines the password associated with the specified **databaseUserId**.
- **databasePortNumber**
  This variable defines the port number where a server is listening for requests from the Java program.
- **databaseServerName**
  This variable defines the name of the database server on which the e*Index database is running. This value can be the name or IP address of the server.
- **databaseVendor**
  This variable defines the database platform being used for the e*Index database. Currently the Java API only supports the Oracle and SQL Server database platforms (enter **Oracle** or **SQL_Server**).
- **databaseName**
  This variable specifies the database to which to connect. This should be the Oracle SID name of the database or, for SQL Server, the ODBC data source name.

- **matchNameServiceId**
  The Vality rule set to use for matching on person names. The default value, **1**, specifies that the default rule set, **UI**, will be used. The file **MatchCfgs.cfg**, located in the **\config** subdirectory, defines and numbers the rule sets used for e*Index. For more information about this file, see "About the Configuration File" later in this chapter.

- **matchAddressServiceId**
  The Vality rule set to use for parsing addresses. The default, **2**, specifies that the United States address rule set will be used. To specify that the Australia address rule set will be used, change this value to **3**; for Great Britain specify **4**; for France specify **5**.

- **traceLevel**
  This variable defines the level of tracing to be performed. The possible values for this variable are TRACE, INFO, WARNING, or ERROR.

- **logicalName**
  This variable defines name for the log files created while running the API. Each EiServer instance must have a unique logical name.

About the Configuration File

The configuration file provided with the Active Integration installation is named **MatchCfgs.cfg**, and is located in **\<home_dir>\config**. This file contains a stanza for each Vality rule set. The **matchNameServiceId** and **matchAddressServiceId** settings in the server properties file specify which of the rule sets listed in the configuration file should be used. The rule sets are specified by number, and the stanzas are numbered in order sequentially with the first stanza being **1**. Following is a sample stanza for a rule set file in **MatchCfgs.cfg**.

```
CONFIG .
KEY UI.DCT
STAN UI.STN
RULES UI.RUL
RECLEN 350
```

The variables included in each stanza are as follows:

- **CONFIG**
  This variable specifies the name of the rule set. By default, the name of the rule set used with e*Index is named **UI**.

- **KEY**
  This variable specifies the name of the match key dictionary file. This file always has an extension of **.DCT**.

- **STAN**
  This variable specifies the name of the standardization file. This file always has an extension of **.STN**.

- **RULES**
  This variable specifies the name of the rules file. This file always has an extension of **.RUL**, and is only located in the stanza for the name-matching rule set.
- **RECLEN**
  This variable specifies the record length for messages processed by the Vality matching algorithm.

Working with the Vality Rule Set Files

The installation of the Java APIs for e*Index Active Integration includes multiple default Vality rule sets for e*Index. The rule set files are read by the Java API at runtime. If you have customized the rule set files being used for e*Index, you need to copy the customized files to the **\<home_dir>\config** directory before you start working with the API set. Unlike the e*Index GUI and e*Ways, these files are not automatically synchronized with the database, so you need to be sure that you are always using the most current version of these files.

About the Sample Files

Sample Java programs are provided with your installation of the Java APIs for e*Index Active Integration in order to illustrate how the components of the API can be used to process and search for data in the e*Index database. The sample files incorporate calls that return control key values, search for UIDs, determine whether a local ID is active, retrieve an active UID given the merged UID, retrieve person records based on demographic information, and add and update person records in the e*Index database. The sample files are all provided in uncompiled **.java** format so you can customize, compile, and run the programs to test them against your e*Index database. For information about specific sample programs and their required arguments, see "Working with the Sample Code" later in this chapter.

Some of the sample files cannot be run in a SQL Server implementation. These sample files include:

- LookupActiveFieldByUser
- LookupActiveFieldBySet
- LookupActiveFunctionByUser
- LookupActiveFunctionBySet
- NextLocalId

About Active Integration Database Tables

Overview

To implement all of the functionality of the Java APIs for e*Index Active Integration in an Oracle environment, you need to install several auxilliary database tables. These tables provide the underlying structure that allow your customized applications to generate local IDs, require that specific fields be entered when adding or updating records, automatically enter default values, and so on. These tables are not available in a SQL Server e*Index implementation.

This section describes each auxilliary table defined for the active API, and also includes information about three tables included in the standard database that are used by the auxilliary tables (*ui_seq_no*, *ui_table*, and *ui_table_column*).

For information about creating the active integration tables, see "Step 7: Create the Active Integration Database Tables" in chapter 7 of the *e*Index Global Identifier Installation Guide*. To see a physical diagram of the auxilliary tables, refer to Figure 3-1 on page 3-11.

ui_seq_no

The *ui_seq_no* is a standard e*Index database table, and is not specific to active integration. This table stores information about the unique, sequential identifier used for certain tables, including the e*Index Active Integration tables. Many columns in the active integration tables refer to this unique identifier rather than the actual object name.

ui_table

The *ui_table* database table lists the tables that store the information that can be displayed on the customized applications. This table is linked to *ui_table_column*, which stores information about each column in the *ui_table* tables. This is a standard e*Index database table, and is not specific to active integration.

ui_table_column

The *ui_table_column* table stores information defining the database columns that store the fields that can be displayed on the customized applications. This table is linked to *ui_table*, and is also a standard e*Index database table.

ui_local_id_generator

The *ui_local_id_generator* table allows you to specify information about the local IDs that are automatically generated by your customized application for each system. Each system for which local IDs will be generated from the Java API must be defined in this table. In this table you define the beginning local ID number, and local IDs are assigned sequentially starting with the number you specify in the seq_no column. When the maximum ID number (max_no) is reached, an exception is issued and local IDs can no longer be generated for the specified system. At this time, the administrator can update the system information and specify a new block of local IDs to use by modifying the **seq_no** and the **max_no** columns

---

Note: *The ID generation is sequential, which means that only numeric characters are allowed in these columns. The warning number functionality (warning_no) is not implemented at this time, but the column must be populated.*

Table 3-1: ui_local_id_generator table

Column Name	Null?	Format	Description
ui_local_id_generator_id	not null	NUMBER(10)	A unique identifier for the record.
facility	not null	VARCHAR2(5)	The facility code of the facility for which you want to generate local IDs from the Java program.
seq_no	not null	NUMBER(25)	The first local ID number to be used by the Java program. If you leave this field null, the API returns a NullPointerException.
warning_no	not null	NUMBER(25)	The local ID number at which a warning will be issued. If you leave this field null, the API returns a NullPointerException.
max_no	not null	NUMBER(25)	The highest local ID number you want generated by the Java program. When this local ID is reached, an exception is issued and local IDs will no longer be generated for this facility. If you leave this field null, the API returns a NullPointerException.
update_date		DATE	The date the record was created or last updated.

ui_active_set

The *ui_active_set* table is specific to active integration and defines a system and setting for each active set. An active set specifies a set of business rules.

An active set, along with a business function, defines a business rules scenario under which certain fields are required, displayed, contain default values, or are automatically filled by trigger values. The chart below describes the columns in this table.

Column Name	Null?	Format	Description
ui_active_set_id	not null	NUMBER(10)	A unique identifier for the active set.
set_code	not null	VARCHAR2(5)	A processing code for the setting in the active set.
facility_code	not null	VARCHAR2(5)	The processing code for the system in the active set.
set_descr		VARCHAR2(30)	A description of the active set.
auto_assign	not null	CHAR(1)	An indicator of whether a local ID is automatically generated for the active set.

ui_active_user_set

The *ui_active_user_set* table defines the users who have access to your customized applications along with the active sets that apply to each user. This combination of data is known as a *user set*. You can limit the information available to users or assign default and trigger values by assigning users to active sets.

Column Name	Null?	Format	Description
ui_active_user_set_id	not null	NUMBER(10)	A unique identifier for the user set.
usersl_id	not null	NUMBER	The user's logon ID.
ui_active_set_id	not null	NUMBER(10)	The unique identifier assigned to the active set associated with the user logon ID.

ui_active_function

The *ui_active_function* table defines the business functions associated with each active set. Each function and active set combination must be unique.

Column Name	Null?	Format	Description
ui_active_function_id	not null	NUMBER(10)	A unique identifier for the function.
ui_active_set_id	not null	NUMBER(10)	The unique identifier assigned to the active set associated with the function.
func_code	not null	VARCHAR2(18)	A processing code for the function.

Column Name	Null?	Format	Description
func_descr		VARCHAR2(30)	A description of the function.

ui_active_field_cfg

The *ui_active_field_cfg* table allows you to indicate which fields on your customized applications are required when records are processed under specific business rules scenarios.

Column Name	Null?	Format	Description
ui_active_field_cfg_id	not null	NUMBER(10)	A unique identifier for the record.
ui_active_function_id	not null	NUMBER(10)	The unique identifier assigned to the function associated with the configuration.
ui_active_field_id	not null	NUMBER(10)	The unique identifier assigned to the active field being defined.
required	not null	CHAR(1)	An indicator of whether the specified field is required.
default_value		VARCHAR2(30)	A default value that automatically appears in the specified field when a window on the custom application is displayed.

ui_active_field

The *ui_active_field* table defines the fields that appear on the customized application along with the name of the field as it appears on your custom applications. The fields listed here must correlate to a data column in one of the database tables listed in *ui_table*.

Column Name	Null?	Format	Description
ui_active_field_id	not null	NUMBER(10)	A unique identifier for the record.
table_column_id	not null	NUMBER(10)	The unique identifier for the column associated with the active field.
field_name	not null	VARCHAR2(30)	The name of the active field.
field_label		VARCHAR2(40)	A label for the active field as you want it to appear on the custom application.

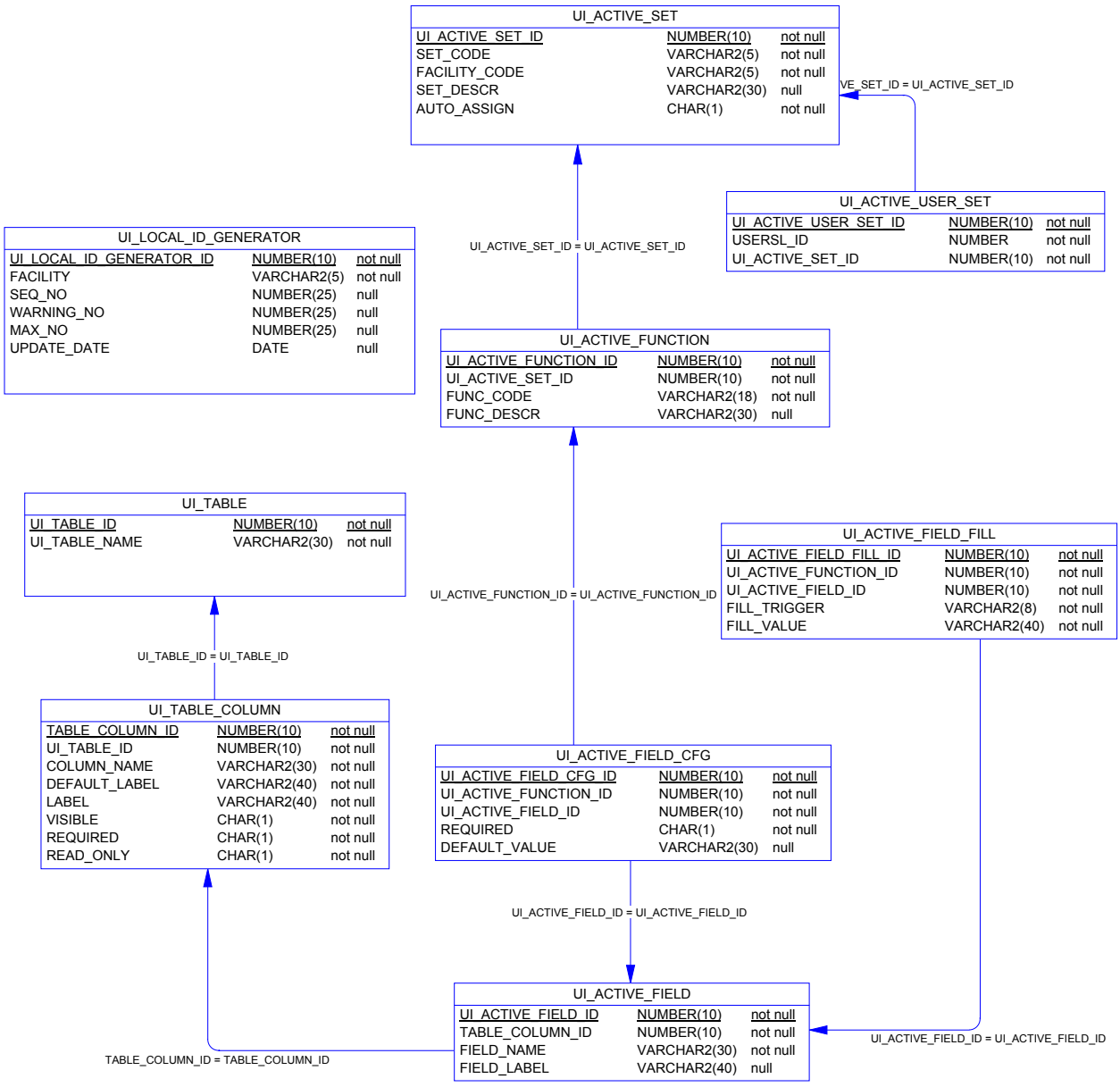
ui_active_field_fill

The *ui_active_field_fill* table allows you to specify the text to be entered automatically in the fields listed in *ui_active_field*. For example, if you specify a fill trigger of "M" with a fill value of "Medicare", then a user only needs to type "M" in the specified field on the custom application and the field will automatically be populated with the text "Medicare".

Column Name	Null?	Format	Description
ui_active_field_fill_id	not null	NUMBER(10)	A unique identifier for the record.
ui_active_function_id	not null	NUMBER(10)	The unique identifier assigned to the function with which the field fill value is associated.
ui_active_field_id	not null	NUMBER(10)	The unique identifier assigned to the field with which the field fill value is associated.
fill_trigger	not null	VARCHAR2(8)	The character that triggers the fill value to be populated.
fill_value	not null	VARCHAR2(40)	The value that is automatically populated into the specified field when the **fill_trigger** is entered.

Figure 3-1: Auxilliary Table Data Model for Oracle

Note that ui_table *and* ui_table_column *are standard database tables. They are included in this diagram to illustrate their relationship to the auxilliary tables.*



Working with the Java API

Overview

This section of the chapter provides information about using Java APIs for e*Index Active Integration and working with the sample files.

Using the Javadocs

In addition to this API reference, documentation for the active integration APIs is provided in Javadoc format. The Javadoc index files are located in the Java API home directory in the **\docs** subdirectory, and the supporting files are located in **\docs\com.stc.eIndex.active.person** and **\docs\com.stc.eIndex.active.exception**. To access the Javadocs, open the file **index.html** in the **\docs** directory. To access an index of all components, open the file **index-all.html**. This page displays a list of all components of the active integration APIs, along with definitions and links for each component.

The Javadocs contain links and cross-references between methods and classes, allowing you to easily browse through the class and method descriptions.

Testing the Installation

You can use the sample code provided with your installation to verify the installation, environment variable set up, and connection to the database. Use the sample **EiServer.properties** file to perform your initial test. To test the installation, perform the following steps:

1. Make sure the paths to the properties file and, for an Oracle environment only, the Oracle JDBC libraries are defined in your **CLASSPATH** variable.
2. Make sure you have an e*Index test database installed and ready to use. Also, make sure that the file **create_id_gen.sql** has been run against the database to create the local ID generator table, and that the new table has been populated with information specific to each system.
3. For Oracle only, if you are defining business rules scenarios, make sure the "active" tables have been created and populated.

Note: *For more information about performing steps 1 through 3, refer to chapter 7 of the* e*Index Global Identifier Installation Guide.

---

4 Modify the **EiServer.properties** file (located in **\<home_dir>\sample**) by entering your database name and server name. Verify the remaining variables to be sure no other changes are required.

5 Execute the **com.stc.eIndex.active.sample.GetControlKeys** class (no parameters required). You should see a listing of the control keys and their current values for the database you specified.

API Class Instantiation

Several of the classes included in the API must be instantiated before calls to the methods defined in those classes can be used. EiServer must be instantiated before any calls are made to the active integration API. Most of the processing logic for the API is provided in the PersonBO class. If you will be searching for, adding, or updating data in the e*Index database, you need to get an instance of PersonBO (this is done by calling **getPersonBOInstance** from the EiBOFactory class). Other classes that must be instantiated include ActiveLookup, EiSystemBO, LocalIdBO, ControlKey, CodeLookup, CountryOptionLookup, LocalIdGenerator, PredefinedMsgRegistry, Security, and ZipCodeLookup.

API Objects for Person Data

Several classes in the active integration API represent person data. The most comprehensive object is the *Person* object, which consists of a *Demographics* object, a *Uid* object, a *Transaction* object, one or more *LocalId* objects, zero or more *Alias* objects, zero or more *Address* objects, zero or more *AuxId* objects, zero or more *Audit* objects, and zero or more *PredefinedMsg* objects. An *Ssn* object represents the social security number in a Demographics object.

Another type of object, *DemographicsRO*, is similar to a Demographics object with the exception that a DemographicsRO object is read-only. Use this type of object if you only want to view person information without performing any edits.

About the PersonBO Class

The PersonBO class is a key class in the API. This class provides the primary methods used for searching, adding, and updating data in the e*Index database. The methods in this class are described in more detail in the following sections, but following is a brief description of the methods and their purposes:

- **searchAlpha**: Performs an alphanumeric search based on a limited set of demographic criteria. The results of this search are weighted.
- **searchGeneral**: Performs an alphanumeric search based on any demographic criteria. The search criteria must contain at least one indexed field. The results of this search are not weighted.

- **searchPhonetic**: Performs a phonetic search based on the specified criteria. The results of this search are weighted.
- **lookupUid**: Searches for the UID or UIDs associated with the social security number or the local ID and system specified.
- **uidLookupByAuxId**: Searches for the UID or UIDs associated with the auxilliary ID object specified.
- **loadPerson**: Populates a person object with a person record from the database using a UID to find the person record.
- **addPerson**: Adds a person record to the database without checking for assumed matches.
- **processPerson**: Adds or updates a person record, based on standard processing logic.
- **updatePerson**: Updates an existing record in the e*Index database.

In addition to these primary functions, PersonBO also provides certain processing logic, including the following:

- **hasLocalId** : Checks whether a local ID already exists in a Person object.
- **hasAlias**: Checks whether an alias already exists in a Person object.
- **hasAddress**: Checks whether a Person object already has an address of the same type.
- **hasAuxId**: Checks whether a non-unique ID already exists in a Person object.
- **hasPhone**: Checks whether a Person object already has a telephone number of the same type.
- **addLocalId**: Adds a LocalId object to a Person object. Call **hasLocalId** first to be sure the record is not added twice.
- **addAlias**: Adds an Alias object to a Person object. Call **hasAlias** first to be sure the record is not added twice.
- **addAddress**: Adds an Address object to a Person object. Call **hasAddress** first to be sure the record is not already associated with an address of the same type.
- **addPhone**: Adds a Phone object to a Person object. Call **hasPhone** first to be sure the record is not already associated with a telephone number of the same type.
- **addAuxId**: Adds an AuxId object to a Person object. Call **hasAuxId** first see if the non-unique ID already exists for a person object.

- **addAudit**: Adds an Audit object to a Person object. Audit information is stored in *ui_audit*, which maintains a record of each instance *ui_person* is accessed.
- **addPredefinedComment**: Adds a PredefinedComment object to a Person object.
- **addUserDefinedComment**: Adds a text comment to a Person object.
- **getActiveUid**: Finds the active UID associated with the specified UID if the specified UID has a status of merged.
- **isActive** (and also **isDeactivated** and **isMerged**): Checks the status of the specified person record.

Processing Searches

The Java APIs for e*Index Active Integration provide a variety of ways to search for person records. You can perform searches and lookups using the following types of criteria:

- UID
- Demographic
- Local ID and System
- Social Security Number
- Non-unique ID and Type

UID

You can use the PersonBO method **loadPerson** to find a person record based on the UID. **loadPerson** looks up the specified UID and then populates a Person object with the information from the person record associated with that UID. Before calling **loadPerson**, you should create a new Person object into which the information is loaded.

Demographic

You can use the PersonBO methods **searchAlpha**, **searchPhonetic**, and **searchGeneral** to perform demographic alphanumeric and phonetic searches against the database. In order to use these calls, you need to create and populate a SearchParameters object to use as the argument for the call to **searchAlpha**, **searchPhonetic**, or **searchGeneral**. Use the setter methods in the Demographics class to populate the SearchParameters object.

When you process a demographic search, a PersonSearchResult object is returned. This object is comprised of DemographicsResultRO objects that contain information about the person records that were returned from the search. You can scroll through the resulting DemographicsResultRO objects using the methods in the PersonSearchResult object. To retrieve a Person

object corresponding to one of the DemographicsResultRO objects, you can isolate the UID by calling **DemographicsResultRO.getUid**, and then calling **loadPerson** using that UID.

Local ID and System

The **lookupUid** method in the PersonBO class allows you to search for a UID based on the specified local ID and system pair. Once you obtain the UID, you can call **loadPerson** to populate a person object with the information corresponding to the UID.

Social Security Number

The **lookupUid** method in the PersonBO class also allows you to search for a UID based on the specified social security number. Once you obtain the UID, you can call **loadPerson** to populate a person object with the information corresponding to the UID.

Non-unique ID and Type

The **uidLookupByAuxId** method in the PersonBO class also allows you to search for a UID based on the specified non-unique ID number and type. Once you obtain the UID, you can call **loadPerson** to populate a person object with the information corresponding to the UID.

Adding Person Data

You can add person records to the e*Index database through two methods: **addPerson** or **processPerson** in the PersonBO class. **addPerson** forces a record to be added without going through the standard processing logic (as defined under "Inbound Event Processing Logic" in chapter 2 of the *e*Index Global Identifier Technical Reference*). **processPerson** uses standard processing logic, and is described in the following section, "Processing Person Data". When you add a person to the database, the UID of the new record is returned.

Before you call **addPerson**, you need to create a new Person object, and populate that object. You can use the setter methods inherited from the Demographics class to populate the demographic properties of the Person object, and the **add*** methods to populate additional information, such as local IDs, alias names, addresses, and so on. You cannot add a new person record without adding a local ID to the Person object.

When you have populated the Person object, calling **addPerson** inserts the new record into the database. One of the parameters taken by **addPerson** is a duplicate check flag. Using this flag, you can specify whether to check for duplicate records. This can be useful when a user needs to enter default information, such as a person record with the name of **John Doe**.

Processing Person Data

To add information to the database using standard e*Index processing logic, you should call **processPerson** instead of **addPerson** (described above). **processPerson** inserts information according to the logic used by the e*Index GUI and e*Ways. Like **addPerson**, you can populate the Person object by using the setter methods in the Demographics class, and the **add*** methods in PersonBO. You must include local ID information in order to process the Person object.

When you call **processPerson** to insert new information, it first performs a search of the database for a local ID record matching the local ID and system specified. If it finds a match, it updates the existing record with the new information. If it does not find a match, it performs matching logic against existing records in the database, and either performs an assumed match or inserts a new person record. Finally, potential duplicate processing for the new record is performed.

Note: *For a complete description of the logic used to process a person record, see "Inbound Event Processing Logic" in chapter 2 of the* e*Index Global Identifier Technical Reference. *This section also describes how certain control key settings affect this process.*

Updating Person Data

The **updatePerson** method is used to perform an update of a person record. Before calling **updatePerson**, you need to create a new Person object and call **loadPerson** to populate the object with the person record you want to update. This means that you need to obtain the UID of the person record you want to modify, so you may need to perform one of the searches listed under "Processing Searches" earlier in order to find the record to update.

After you load the person record, you can modify the information using the setter methods in the Demographics class. You can also add alias, local ID, address, phone, non-unique ID, and audit information using the **add*** methods in PersonBO, but you should call the **has*** methods (**hasAlias**, **hasLocalId**, **hasAddress**, and so on) to be sure that you don't add duplicate information to the person record (see the sample **UpdatePerson.java** for an example of how to use **hasLocalId**). **updatePerson** returns a Boolean value indicating whether the update was successful.

Customizing the Application Appearance

The Java APIs for e*Index Active Integration include a set of classes that you can use to define the appearance of your customized applications under different scenarios. These classes use the active integration tables described under "About Active Integration Database Tables" earlier in this chapter, and are only available to Oracle implementations. These classes are: **ActiveField**,

ActiveFieldFill, **ActiveFunction**, **ActiveLookup**, and **ActiveSet**. Using these classes, you can define required fields, field labels, default values, and fill/trigger values for business rules scenarios.

A business rules scenario consists of a setting, a system, and a business function. The setting and system combinations you define are stored in *ui_active_set*, and are known as *active sets*. Each active set can be associated with several business functions, and each function can be associated with several field definitions. You can also specify whether local IDs are automatically assigned for each active set. Finally, you can assign business rules scenarios to specific users, giving each user a version of the application that is precisely customized for their processing needs.

Working with the Sample Code

There are several sample **.java** files included in the sample directory in the API installation. These files provide examples of how the classes and methods can be combined to create a program that meets your functional requirements. Once you set up your Java environment and set all the environment variables, you can run these programs to see how they manipulate data contained in the e*Index database and process data into the database. For information about setting environment variables for the active integration API, see chapter 7 of the *e*Index Global Identifier Installation Guide*. Following are descriptions of the sample programs.

AddPerson

AddPerson provides an example of how to add a new person record to the database using **addPerson** in the PersonBO class. This sample also adds a local ID, alias, address, telephone number, non-unique ID, and audit record to the Person object. The sample returns the UID associated with the new person record. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.AddPerson
```

There are several user-defined variables within this sample that you can modify to suit your data configuration.

GetActiveUid

GetActiveUid provides an example of how to find the active UID associated with a merged UID using the method **getActiveUid** in the PersonBO class. This function is used to ensure that you don't update a record that has been merged into another record and is no longer active. The sample returns the active UID associated with the UID you specify as the argument. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.GetActiveUid uid
```

where ***uid*** is the merged UID for which you want to find the active UID.

GetAllSystems

GetAllSystems provides an example of how to retrieve an enumeration of system records and then scroll through each record in the enumeration. This sample prints the system code, description, and status for each system you have defined. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.GetAllSystems
```

GetControlKeys

GetControlKeys provides an example of how to look up the values of the e*Index control keys using the methods in the ControlKey class. This sample returns a listing of e*Index control keys and their associated values. It requires no parameters. This syntax for this sample program is:

```
com.stc.eIndex.active.sample.GetControlKeys
```

GetPerson

GetPerson provides an example of how to use **loadPerson** in the PersonBO class to retrieve a person record from the database given the UID associated with the record. The syntax of this sample program is:

```
com.stc.eIndex.active.sample.GetPerson uid
```

where ***uid*** is the UID of the person record you want to retrieve. If the record is retrieved successfully, this sample displays a string representation of the Person object.

GetSystem

GetSystem provides an example of how to use **getSystem** in the EiSystemBO class to retrieve a system record from the database given the processing code associated with the record. The syntax of this sample program is:

```
com.stc.eIndex.active.sample.GetSystem system_code
```

where ***system_code*** is the processing code of the system record you want to retrieve. If the record is retrieved successfully, this sample displays the value of each field in the System object.

IsUserValid

IsUserValid provides an example of how to use **isUserValid** in the Security class to check the *user_tbl* for valid login IDs. This program checks the table to see if the specified user login ID exists and is currently active in the table. If the user ID exists and is active, the program returns the string **VALID USER**; if the user ID does not exist or is inactive, the program returns the string **INVALID USER**. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.IsUserValid user_id
```

where ***user_id*** is the login ID to check. The parameter is case-sensitive.

LookupActiveFieldBySet

LookupActiveFieldBySet is only available in an Oracle implementation, and provides an example of how to use **getActiveFields** in the ActiveLookup class to retrieve an enumeration of active field definitions. This sample uses methods from the ActiveField, ActiveFieldFill, and ActiveLookup classes. If the records are retrieved successfully, this sample displays active field information (such as field labels, whether a field is required, default values, and so on) as well as fill trigger and fill value information for each field. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupActiveFieldBySet set_code
function_code
```

where ***set_code*** is the active set code and ***function_code*** is the business function code you want to look up. These parameters are case sensitive.

LookupActiveFieldByUser

LookupActiveFieldByUser is only available in an Oracle implementation, and provides an example of how to use **getActiveFieldsByUser** in the ActiveLookup class to retrieve an enumeration of active field definitions for the specified user and function. This sample uses methods from the ActiveField, ActiveFieldFill, and ActiveLookup classes. If the records are retrieved successfully, this sample displays active field information (such as field labels, whether a field is required, default values, and so on) as well as fill trigger and fill value information for each field. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupActiveFieldByUser user_id
function_code
```

where ***user_id*** is the user login ID and ***function_code*** is the business function code you want to look up. These parameters are case sensitive.

LookupActiveFunctionBySet

LookupActiveFunctionBySet is only available in an Oracle implementation, and provides an example of how to use **getActiveFunctions** in the ActiveLookup class to retrieve an enumeration of active functions defined for the specified active set. This sample uses methods from the ActiveFunction and ActiveLookup classes. If the records are retrieved successfully, this sample displays the function code and description for each function associated with the specified active set. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupActiveFunctionBySet set_code
```

where ***set_code*** is the active set code you want to look up. This parameter is case sensitive.

LookupActiveFunctionByUser

LookupActiveFunctionByUser is only available in an Oracle implementation, and provides an example of how to use **getActiveSet** in the ActiveLookup class to retrieve active set information for the specified user. The sample obtains the set code from the returned active set, and then calls **getActiveFunctions** using that set code. If the records are retrieved successfully, this sample displays the function code and description for each function associated with the specified user ID. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupActiveFunctionByUser user_id
```

where ***user_id*** is the user login ID of the person whose active functions you want to look up. This parameter is case sensitive.

LookupCode

LookupCode provides an example of how to use the fields in the EnumCodeType class and the methods in CodeLookup to find the display description for a specified processing code. For example, if you specify a code type of **gender** and a code of **F**, this program returns **FEMALE**. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupCode codetype code
```

where ***codetype*** is the processing code type and ***code*** is the processing code for which you want to find the display value. These parameters are case sensitive.

LookupUidByLocalId

LookupUidByLocalId provides an example of how to use **lookupUid** to find a UID based on a system and local ID pair. You can then use the returned UID to load a person record. This sample returns the UID that is associated with the local ID and system code you specify. The context for this sample program is:

```
com.stc.eIndex.active.sample.LookupUidByLocalId system lid
```

The ***system*** argument is the system code of the system, and is case sensitive. The ***lid*** argument is the local identifier

LookupUidBySsn

LookupUidBySsn provides an example of how to use **lookupUid** to find a UID based on a social security number. You can then use the UID to load a person record. This sample returns a UID or list of UIDs associated with the social security number you specify. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.LookupUidBySsn ssn
```

where ***ssn*** is the social security number of the record you want to find.

LookupZipCode

LookupZipCode provides an example of how to use the methods in the ZipCodeLookup and CityState classes to find the city and state associated with the zip code and, optionally, the zip code extension you specify. This sample returns a list of city and state pairs. The syntax for this program is:

```
com.stc.eIndex.active.sample.LookupZipCode zipcode zipext
```

where ***zipcode*** is the 5-digit zip code you want to look up and ***zipext*** is the associated zip code extension. If you do not want to specify a zip code extension, enter a percent sign (%) for the ***zipext*** parameter.

NextLocalId

NextLocalId is only available in an Oracle implementation, and provides an example of how to use the methods in the LocalIdGenerator class to obtain the next available local ID for the specified system (in this sample the default system is **SBYN**). The next available ID is specified in the database table *ui_local_id_generator*, and the system **SBYN** must be defined in the table in order for this sample to work. If you define additional systems in *ui_local_id_generator*, you can modify the file **NextLocalId.java** to generate local IDs for different systems. This program does not accept any parameters. The syntax for this program is:

```
com.stc.eIndex.active.sample.NextLocalId system_code
```

where ***system_code*** is the processing code for the system whose next local ID you want to look up.

ParameterValidator

Several of the other sample files use this file to validate and define the arguments you need to specify when you execute each sample program. It does not use any of the Java functions specific to e*Index.

PersonSearch

PersonSearch provides an example of how you can use methods in the PersonSearch, SearchParameters, and PersonBO classes to perform an alphanumeric or phonetic demographic search. This sample performs a search based on criteria you specify (in a SearchParameters object) and returns a list of matching records (in a PersonSearchResult object). The syntax for this sample program is:

```
com.stc.eIndex.active.sample.PersonSearch search_type fname
lname mname gender ssn dob
```

where ***search_type*** is either **A** (for alphanumeric) or **P** (for phonetic) and the remaining arguments are the search criteria as follows: ***fname*** is the first name, ***lname*** is the last name, ***mname*** is the middle name, ***gender*** is the gender, ***ssn*** is the social security number, and ***dob*** is the date of birth. To leave any of the search criteria blank, enter a percent sign (%) in its place.

ProcessPerson

ProcessPerson provides an example of how you can use methods in the Person, PersonBO, LocalId, and Transaction classes to add information to the database. This sample uses the Demographics setter methods to populate a Person object, and then adds a local ID to the Person object. It then creates a Transaction object, and calls **loadPerson** to either insert a new person record or update an existing one. The demographic and local ID values to process are defined directly in this program. Be sure to modify these values if you want to run this file multiple times. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.ProcessPerson
```

RunGeneralSearch

RunGeneralSearch provides an example of how you can use methods in the PersonBO, SearchParameters, and PersonSearchResult classes to perform a general search. This sample uses a special function to load the search criteria that are defined in the file **GeneralSearch.Properties**. Modify the parameter values in **GeneralSearch.Properties** (located in the Java API home directory in **/sample**) to search on any combination of data. At least one search field must be an indexed field. For any field you do not want to use as search criteria, leave the value blank. If using common code table elements as criteria (such as race, religion, language, and so on), enter the processing code and not the item description. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.RunGeneralSearch
```

UpdatePerson

UpdatePerson provides an example of how you can use methods in the Person, Person BO, LocalId, and Transaction classes to update a person record in the database. This sample uses the Demographics setter methods to modify demographic fields. It then checks if the specified local ID record already exists in the person record, and if not, adds the local ID to the Person object. It then creates a Transaction object and calls **updatePerson** to update the specified record. The demographic and local ID values to process are defined directly in this program. Be sure to modify these values if you want to run this file multiple times. Also, make sure the UID specified in the call to **loadPerson** is a valid active UID in the e*Index database. The syntax for this sample program is:

```
com.stc.eIndex.active.sample.UpdatePerson
```

For More Information

Other SeeBeyond publications may help you to learn how to perform tasks associated with creating Java programs for e*Index.

To learn more about …	See …
The search parameters and search functions of e*Index	Chapter 3 of the *e*Index Global Identifier User Guide*
The standard search results set	Chapter 3 of the *e*Index Global Identifier User Guide*
Adding and updating person records in e*Index	Chapter 4 of the *e*Index Global Identifier User Guide*
e*Index Control Keys	The *e*Index Data Dictionary User Guide*

Chapter 4

Person Package

About this Chapter

Overview

This chapter provides a complete listing and descriptions of the Java APIs included in the **com.stc.eIndex.active.person** package for e*Index. It also includes examples of how to use many of the methods included in the Person package.

The following diagram illustrates each major topic in this chapter.

Learn about the Person package provided in the Java APIs for e*Index Active Integration

Learn about the implementation, syntax, and parameters of the classes and methods belonging to the Person package

About the Person Package

Overview

This section of the chapter provides the background information you should know before using the classes and methods provided in the package **com.stc.eIndex.active.person**.

The Person Package

The Person package includes the classes and methods you need to implement in order to search for and process person data in the e*Index database. Classes in this package provide a variety of functions, including database connectivity, system configuration, searching for person records, adding person records, updating person records, looking up processing codes, and so on. For a complete list of the classes, fields, and methods included in the Person package, see Table 2-1 beginning on page 2-6. For information about how the components of the Person package can be combined to process e*Index data, see "Working with the Java API" in chapter 3 of this guide.

For More Information

Other SeeBeyond publications may help you to learn how to perform tasks associated with configuration APIs.

To learn more about …	See …
Control Keys	Chapter 5 of the e*Index Administrator User's Guide
Standard e*Index data processing	Chapter 2 of the e*Index Global Identifier Technical Reference
e*Index database tables	Chapter 2 of the e*Index Global Identifier Technical Reference
Standard e*Index GUI functions	Chapters 3 and 4 of the e*Index Global Identifier User's Guide
Country-specific Options	Chapter 5 of the e*Index Administrator User's Guide

ActiveField Class

Description

The **ActiveField** class is used by the ActiveLookup class to return a set of ActiveField objects. This class represents information about the fields for specific business rules scenarios, such as the fields that are required, default values for certain fields, field labels, and so on. ActiveField also represents information about the database tables and columns associated with the defined fields. The methods in this class are only available in Oracle implementations.

Properties

The **ActiveField** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
|
└ com.stc.eIndex.active.person.ActiveField
```

Constructor

None

Methods

The methods included in the **ActiveField** class are described in detail on the following pages:

Inherited Methods

The **ActiveField** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getActiveFieldFillEnumeration

Description

The **getActiveFieldFillEnumeration** method retrieves an enumeration of the field fill objects for an ActiveField object. Use this method to retrieve the fill triggers and fill values defined for a field in an ActiveField object.

Syntax

```
public java.util.Enumeration getActiveFieldFillEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getActiveFieldFillEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration	The enumeration of the active field fill objects was retrieved successfully.

Throws

None.

Example

The following example calls **getActiveFields** to display a list of ActiveField objects associated with the specified *set_ code* and *function_code*. The '**get**' methods in the ActiveField class are called to display information about the fields defined for the specified setting and function, and then **getActiveFieldFillEnumeration** is called to retrieve an enumeration of field fill definitions for each ActiveField object. Finally, '**get**' methods in the ActiveFieldFill class are called to display field trigger and fill information.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 Enumeration eaf, eff;
 eaf = activeLookup.getActiveFields(set_code, function_code);
 while (eaf.hasMoreElements()) {
     ActiveField af = (ActiveField)eaf.nextElement();
     System.out.println("Field: " + af.getFieldName());
     System.out.println("Label: " + af.getFieldLabel());
     System.out.println("Required: " + af.getRequired());
     System.out.println("Default: " + af.getDefaultValue());
     System.out.println("Table: " + af.getTableName());
     System.out.println("Database Column: " + af.getDbColumnName());
     System.out.println("Function Code: " + af.getFuncCode());
     eff = af.getActiveFieldFillEnumeration();
     if(eff != null) {
       while (eff.hasMoreElements()) {
         ActiveFieldFill ff = (ActiveFieldFill)eff.nextElement();
         System.out.println("Fill trigger: " + ff.getFillTrigger());
         System.out.println("Fill value: " + ff.getFillValue());
       }
     }
 }
...
```

getDbColumnName

Description

The **getDbColumnName** method retrieves the name of the database column associated with the fields represented in an ActiveField object.

Syntax

```
public java.lang.String getDbColumnName()
```

Parameters

Parameter	Description
None	

Return Value

The **getDbColumnName** method returns the following value:

This value is returned …	if this occurs …
String containing a database column name	The value of the **dbColumnName** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getDbColumnName** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getDefaultValue

Description

The **getDefaultValue** method retrieves the value of the default value for the field represented in an ActiveField object. Use this method to automatically populate specific values into a field when a browser window appears.

Syntax

```
public java.lang.string getDefaultValue()
```

Parameters

Parameter	Description
None	

Return Value

The **getDefaultValue** method returns one of the following values:

This value is returned …	if this occurs …
String containing the specified default value	The value of the **defaultValue** field for the ActiveField object was retrieved successfully.
Null	The **defaultValue** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getDefaultValue** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getFieldLabel

Description

The **getFieldLabel** method retrieves the name of the field label for the field represented by the ActiveField object. Use this method to display the label for a specific field.

Syntax

```
public java.lang.String getFieldLabel()
```

Parameters

Parameter	Description
None	

Return Value

The **getFieldLabel** method returns one of the following values:

This value is returned …	if this occurs …
String containing a field label name	The value of the **fieldLabel** field was retrieved successfully.
Null	The **fieldLabel** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getFieldLabel** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getFieldName

Description

The **getFieldName** method retrieves the default name of the field that is represented by the ActiveField object.

Syntax

```
public java.lang.String getFieldName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFieldName** method returns the following value:

This value is returned …	if this occurs …
String containing the name of a field	The value of the **fieldName** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getFieldName** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getFuncCode

Description

The **getFuncCode** method retrieves the function code of the function specified for an ActiveField object. Use this method to retrieve the function with which the business rules scenario should be used.

Syntax

```
public java.lang.String getFuncCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getFuncCode** method returns the following value:

This value is returned …	if this occurs …
String containing a function code	The value of the **funcCode** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getFuncCode** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getRequired

Description

The **getRequired** method retrieves the value of the **required** property in an ActiveField object. Use this method to determine whether a field must be filled in when performing a function for a specific business rules scenario.

Syntax

```
public java.lang.String getRequired()
```

Parameters

Parameter	Description
None	

Return Value

The **getRequired** method returns the following value:

This value is returned …	if this occurs …
String containing an indicator of whether a field is required	The value of the **required** field was retrieved successfully. The return value will be either **Y** (required) or **N** (not required).

Throws

None.

Example

To see an example of how **getRequired** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getSetCode

Description

The **getSetCode** method retrieves the value of the **setCode** property. Use this method to retrieve the setting defined for the business rules scenario.

Syntax

```
public java.lang.String getSetCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSetCode** method returns the following value:

This value is returned …	if this occurs …
String containing the setting code value	The value of the **setCode** property was retrieved successfully.

Throws

None.

Example

To see an example of how **getSetCode** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

getTableName

Description

The **getTableName** method retrieves the name of the database table that contains the field represented in an ActiveField object.

Syntax

```
public java.lang.String getTableName()
```

Parameters

Parameter	Description
None	

Return Value

The **getTableName** method returns the following value:

This value is returned …	if this occurs …
String containing a table name	The name of the table was retrieved successfully.

Throws

None.

Example

To see an example of how **getTableName** in the ActiveField class can be used, see the example code provided for **getActiveFieldFillEnumeration** on page 4-5.

ActiveFieldFill Class

Description

The **ActiveFieldFill** class is used by the ActiveField class to retrieve an enumeration of field trigger and field fill definitions. Field triggers and fills are used to automatically populate a field when the specified trigger character is entered into the field. The methods in this class are only available to Oracle implementations.

Properties

The **ActiveFieldFill** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
|
└ com.stc.eIndex.active.person.ActiveFieldFill
```

Constructor

None

Methods

The methods included in the **ActiveFieldFill** class are described in detail on the following pages:

- **getFieldName** on page 4-14
- **getFillTrigger** on page 4-15
- **getFillValue** on page 4-16
- **getFunctionCode** on page 4-17

Inherited Methods

The **ActiveFieldFill** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getFieldName

Description

The **getFieldName** method retrieves the value of the **fieldName** property in an ActiveFieldFill object. Use this method to display the name of the field for which the triggers and fills are defined.

Syntax

```
public java.lang.String getFieldName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFieldName** method returns the following value:

This value is returned …	if this occurs …
String containing a field name	The value of the **fieldName** field was retrieved successfully.

Throws

None.

Example

The following example calls **getActiveFields** to display a list of ActiveField objects associated with the specified *set_ code* and *function_code*. **getActiveFieldFillEnumeration** is called to retrieve an enumeration of field fill definitions for each ActiveField object. Finally, the '**get**' methods in the ActiveFieldFill class are called to display field trigger and fill information.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 Enumeration eaf, eff;
 eaf = activeLookup.getActiveFields(set_code, function_code);
 while (eaf.hasMoreElements()) {
     ActiveField af = (ActiveField)eaf.nextElement();
     if(eff != null) {
       eff = af.getActiveFieldFillEnumeration();
       while (eff.hasMoreElements()) {
         ActiveFieldFill ff = (ActiveFieldFill)eff.nextElement();
         System.out.println("Field Name: " + ff.getFieldName());
         System.out.println("Function Code: " + ff.getFunctionCode));
         System.out.println("Fill trigger: " + ff.getFillTrigger());
         System.out.println("Fill value: " + ff.getFillValue());
       }
     }
 }
...
```

getFillTrigger

Description

The **getFillTrigger** method retrieves the **fillTrigger** property. Use this method to display the character that, when entered into a field, triggers a fill value to be populated into that field.

Syntax

```
public java.lang.String getFillTrigger()
```

Parameters

Parameter	Description
None	

Return Value

The **getFillTrigger** method returns the following value:

This value is returned ...	if this occurs ...
String containing a fill trigger	The value of the **fillTrigger** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getFillTrigger** in the ActiveFieldFill class can be used, see the example code provided for **getFieldName** beginning on page 4-14.

getFillValue

Description

The **getFillValue** method retrieves the fill value property. Use this method to retrieve a string that is automatically populated into a field when the trigger character is entered into that field.

Syntax

```
public java.lang.String getFillValue()
```

Parameters

Parameter	Description
None	

Return Value

The **getFillValue** method returns the following value:

This value is returned …	if this occurs …
String containing a fill value	The value of the **fillValue** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getFillValue** in the ActiveFieldFill class can be used, see the example code provided for **getFieldName** beginning on page 4-14.

getFunctionCode

Description

The **getFunctionCode** method retrieves the function code in an ActiveFieldFill enumeration. Use this method to display the function for which a specific fill value and trigger are used.

Syntax

```
public java.lang.String getFunctionCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getFunctionCode** method returns the following value:

This value is returned …	if this occurs …
String containing a function code	The value of the **functionCode** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getFunction Code** in the ActiveFieldFill class can be used, see the example code provided for **getFieldName** beginning on page 4-14.

ActiveFunction Class

Description

The **ActiveFunction** class is used by the ActiveLookup class to retrieve information about the functions defined for business rules scenarios. A business rule scenario is defined by setting, system, and function. The methods in this class are only available to Oracle implementations.

Properties

The **ActiveFunction** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└ com.stc.eIndex.active.person.ActiveFunction

Constructor

None

Methods

The methods included in the **ActiveFunction** class are described in detail on the following pages:

- **getFunctionCode** on page 4-19
- **getFunctionDescription** on page 4-20
- **getSetCode** on page 4-20

Inherited Methods

The **ActiveFunction** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getFunctionCode

Description

The **getFunctionCode** method retrieves the value of the **functionCode** property. Use this method to display the function code associated with a specific business rules scenario.

Syntax

```
public java.lang.String getFunctionCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getFunctionCode** method returns the following value:

This value is returned …	if this occurs …
String containing a function code	The value of the **functionCode** field was retrieved successfully.

Throws

None.

Example

The following example obtains an instance of the ActiveLookup class and then calls **getActiveFunctions** to retrieve an enumeration of the functions defined for the specified setting (defined here by the variable *set_code*). The '**get**' methods in ActiveFunction are called to display information about the defined functions.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 Enumeration eaf = activeLookup.getActiveFunctions(set_code);
   while (eaf.hasMoreElements()) {
     ActiveFunction af = (ActiveFunction)eaf.nextElement();
     System.out.println("Function: " + af.getFunctionCode());
     System.out.println("Description: " + af.getFunctionDescription());
     System.out.println("Setting Code:  " + af.getSetCode());
 }
...
```

getFunctionDescription

Description

The **getFunctionDescription** method retrieves the value of the **functionDescription** property. Use this method to display descriptions of the functions defined for a setting.

Syntax

```
public java.lang.String getFunctionDescription()
```

Parameters

Parameter	Description
None	

Return Value

The **getFunctionDescription** method returns one of the following values:

This value is returned …	if this occurs …
String containing a function description	The value of the **functionDescription** field was retrieved successfully.
Null	The **functionDescription** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getFunctionDescription** in the ActiveFunction class can be used, see the example code provided for **getFunctionCode** beginning on page 4-19.

getSetCode

Description

The **getSetCode** method retrieves the value of the **setCode** property. Use this method to display the setting code associated with a function.

Syntax

```
public java.lang.String getSetCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSetCode** method returns the following value:

This value is returned …	if this occurs …
String containing a setting code	The value of the **setCode** field was retrieved successfully.

Throws

None.

Example

To see an example of how **getSetCode** in the ActiveFunction class can be used, see the example code provided for **getFunctionCode** beginning on page 4-19.

ActiveLookup Class

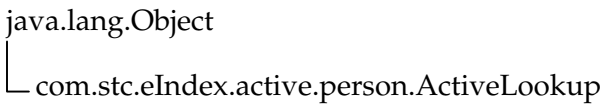
Description

The **ActiveLookup** class looks up sets, functions, and fields that define the business rules scenarios you can use to specify default values, required fields, and field fill values. The methods in this class are only available to Oracle implementations.

Properties

The **ActiveLookup** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None

Methods

The methods included in the **ActiveLookup** class are described in detail on the following pages:

- **getActiveFields** on page 4-23
- **getActiveFieldsByUser** on page 4-24
- **getActiveFunctions** on page 4-25
- **getActiveSet** on page 4-26
- **getInstance** on page 4-27

Inherited Methods

The **ActiveLookup** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getActiveFields

Description

The **getActiveFields** method retrieves the fields defined for the given setting and function. It returns an enumeration of ActiveField objects.

Syntax

```
public java.util.Enumeration getActiveFields(java.lang.String
set, java.lang.String function)
```

Parameters

Parameter	Description
set	The setting for which you are looking up field-specific information.
function	The function for which you are looking up field-specific information.

Return Value

The **getActiveFields** method returns the following value:

This value is returned ...	if this occurs ...
An enumeration of active field objects	The active fields for the given setting and function were retrieved successfully.

Throws

The **getActiveFields** method throws the following exception:

- **EiException Class**
- **java.sql.SQLException**

Example

The following example obtains an instance of ActiveLookup and then calls **getActiveFields** to display a list of ActiveField objects associated with the specified *set_ code* and *func_code*.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 Enumeration eaf = activeLookup.getActiveFields(set_code, func_code);
 while (eaf.hasMoreElements()) {
     ActiveField af = (ActiveField)eaf.nextElement();
     /* Displaying active field information */
 }
...
```

getActiveFieldsByUser

Description

The **getActiveFieldsByUser** method retrieves the active fields for a specified user ID and function. It returns an enumeration of ActiveField objects.

Syntax

```
public java.util.Enumeration
getActiveFieldsByUser(java.lang.String userId, java.lang.String
function)
```

Parameters

Parameter	Description
userId	The logon user ID for which you want to look up field information.
function	The functions associated with the logon ID for which you want to look up field information.

Return Value

The **getActiveFieldsByUser** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration of active field objects	The active fields for the specified user ID and function were retrieved successfully.

Throws

The **getActiveFieldsByUser** method throws the following exception:

- **EiException Class**
- **java.sql.SQLException**

Example

The following example obtains an instance of ActiveLookup and then calls **getActiveFieldsByUser** to display a list of ActiveField objects associated with the specified *user_id* and *func_code*.

```
...
EiServer eiServer = new EiServer("EiServer.properties");
ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

Enumeration eaf, eff;
eaf = activeLookup.getActiveFieldsByUser(user_id, func_code);
while (eaf.hasMoreElements()) {
  ActiveField af = (ActiveField)eaf.nextElement();...
    /* Displaying active field information */
```

getActiveFunctions

Description

The **getActiveFunctions** method retrieves the functions defined for the specified setting. It returns an enumeration of ActiveFunction objects.

Syntax

```
public java.util.Enumeration
getActiveFunctions(java.lang.String set)
```

Parameters

Parameter	Description
set	The setting code for which you want to look up functions.

Return Value

The **getActiveFunctions** method returns one of the following values:

This value is returned …	if this occurs …
An enumeration of active functions	The active functions for the specified setting were retrieved successfully.

Throws

The **getActiveFunctions** method throws the following exception:

- **EiException Class**
- **java.sql.SQLException**

Example

The following example obtains an instance of the ActiveLookup class and then calls **getActiveFunctions** to retrieve an enumeration of the functions defined for the specified setting (defined here by the variable *set_code*). The '**get**' methods in ActiveFunction are called to display information about the defined functions.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 Enumeration eaf = activeLookup.getActiveFunctions(set_code);
   while (eaf.hasMoreElements()) {
     ActiveFunction af = (ActiveFunction)eaf.nextElement();
       /*  Displaying function information  */
 }
...
```

getActiveSet

Description

The **getActiveSet** method retrieves the active setting defined for the specified user ID.

Syntax

```
public ActiveSet getActiveSet(java.lang.String userId)
```

Parameters

Parameter	Description
userId	The user ID for which you want to look up the active setting.

Return Value

The **getActiveSet** method returns one of the following values:

This value is returned …	if this occurs …
An ActiveSet object	The active setting for the specified user ID was retrieved successfully.

Throws

The **getActiveSet** method throws the following exception:

- **EiException Class**
- **java.sql.SQLException**

Example

The following example obtains an instance of ActiveLookup and then calls **getActiveSet** for the specified user ID (*user_id*). It then uses the setting code from the returned ActiveSet object to retrieve an enumeration of active functions.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 ActiveSet as = activeLookup.getActiveSet(user_id);
 String set = as.getSetCode();

 Enumeration eaf = activeLookup.getActiveFunctions(set);
...
```

getInstance

Description

The **getInstance** method retrieves a single instance of the ActiveLookup class for the specified EiServer object, maintaining a singleton pattern so database results are stored for servicing subsequent lookups

Syntax

```
public static ActiveLookup getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the ActiveLookup instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of ActiveLookup	An ActiveLookup instance for the specified EiServer object was retrieved successfully.

Throws

The **getInstance** method throws the following exception:

- **java.sql.SQLException**

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

To see an example of how **getInstance** in the ActiveFunction class can be used, see the example code provided for **getActiveSet** beginning on page 4-26.

ActiveSet Class

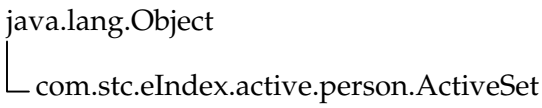
Description

The **ActiveSet** class is used by the ActiveLookup class to retrieve information about the settings defined for business rules scenarios. A business rule scenario is defined by setting, system, and function. The methods in this class are only available to Oracle implementations.

Properties

The **ActiveSet** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None

Methods

The methods included in the **ActiveSet** class are described in detail on the following pages:

- **getAutoAssign** on page 4-29
- **getSetCode** on page 4-30
- **getSetDescription** on page 4-30
- **getSystemCode** on page 4-31

Inherited Methods

The **ActiveSet** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getAutoAssign

Description

The **getAutoAssign** method retrieves the **autoAssign** property for an ActiveSet object. This property indicates whether local IDs are automatically generated when a record is added in a defined business rules scenario.

Syntax

```
public java.lang.String getAutoAssign()
```

Parameters

Parameter	Description
None	

Return Value

The **getAutoAssign** method returns one of the following values:

This value is returned ...	if this occurs ...
String containing a character	The value of the **autoAssign** field was retrieved successfully.

Throws

None.

Example

The following example obtains an instance of ActiveLookup and then calls **getActiveSet** to retrieve the setting defined for the specified user ID (*user_id*). It then calls the '**get**' methods in the ActiveSet class to retrieve information about the setting that was retrieved.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 ActiveLookup activeLookup = ActiveLookup.getInstance(eiServer);

 ActiveSet as = activeLookup.getActiveSet(user_id);
    System.out.println("Setting Code:  "  + as.getSetCode());
    System.out.println("Setting Desc:  "  + as.getSetDescription());
    System.out.println("System Code:   "  + as.getSystemCode());
    System.out.println("Auto Assign?:  "  + as.getAutoAssign());
...
```

getSetCode

Description

The **getSetCode** method retrieves the **setCode** property in an ActiveSet object. Use this field to retrieve the setting code in a business rules scenario.

Syntax

```
public java.lang.String getSetCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSetCode** method returns the following value:

This value is returned …	if this occurs …
String containing setting code	The value of the **setCode** field was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods in the ActiveSet class can be used, see the example code provided for **getAutoAssign** beginning on page 4-29.

getSetDescription

Description

The **getSetDescription** method retrieves the **setDescription** field for an ActiveSet object. Use this method to display a description for a setting in a business rules scenario.

Syntax

```
public java.lang.String getSetDescription()
```

Parameters

Parameter	Description
None	

Return Value

The **getSetDescription** method returns the following value:

This value is returned …	if this occurs …
String containing set description	The value of the **setDescription** field was retrieved successfully.
Null	The **setDescription** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the ActiveSet class can be used, see the example code provided for **getAutoAssign** beginning on page 4-29.

getSystemCode

Description

The **getSystemCode** method retrieves the **systemCode** property in an ActiveSet object. Use this method to retrieve a system code defined in a business rules scenario.

Syntax

```
public java.lang.String getSystemCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSystemCode** method returns the following value:

This value is returned …	if this occurs …
String containing a system code	The value of the **systemCode** field was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods in the ActiveSet class can be used, see the example code provided for **getAutoAssign** beginning on page 4-29.

Address Class

Description

The **Address** class represents address information associated with a person record. Use the methods in this class to create Address objects and to populate or retrieve the fields in an Address object.

Properties

The **Address** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**

java.lang.Object
└ com.stc.eIndex.active.core.DataObject
└ com.stc.eIndex.active.person.PersonDependent
└ com.stc.eIndex.active.person.Address

Constructor

The **Address** class has one constructor, which is described on the following page:

- **Address** on page 4-35

Methods

The methods included in the **Address** class are described in detail on the following pages:

- **equals** on page 4-36
- **getAddress1** on page 4-37
- **getAddress2** on page 4-39
- **getAddress3** on page 4-39
- **getAddress4** on page 4-40
- **getAddressType** on page 4-41
- **getCity** on page 4-42
- **getCountry** on page 4-43
- **getCounty** on page 4-43
- **getHouseNumber** on page 4-44
- **getPostalCode** on page 4-45
- **getPostalCodeExt** on page 4-46
- **getStateOrProvince** on page 4-47

Inherited Methods

The **Address** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Address

Description

The **Address** method is the constructor method for the Address class. Use this method to create objects containing a person's address information.

Syntax

```
public Address(java.lang.String type)
```

Parameters

Parameter	Description
type	The processing code for the type of address you are adding to the person record, such as **H** (home), **O** (office), and so on.

Return Value

The **Address** constructor method returns the following value:

This value is returned …	if this occurs …
An object containing address information	The Address object was created successfully.

Throws

The **Address** constructor throws the following exception:

- **EiException Class**

Example

The following example is excerpted from a sample used to create a new Person object and then add demographic, address, and local ID information to the new object. It obtains a new instance of EiServer and PersonBO, and then creates an empty Person object. It then creates a Demographics object to populate the demographic fields of the Person object, and an empty Address object **address** to populate the address information. The '**set**' methods are called to fill the Address object with address information, and **addAddress** is called to add the Address object to the Person object.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();

 Demographics demo = new Demographics();
     demo.setLastName(last_name);
     demo.../* Populating property values */
 ...
 Address address = new Address("H");
     address.setAddress1(address_street1);
     address.setAddress2(address_street2);
     address.setAddress3(address_street3);
     address.setAddress4(address_street4);
     address.setCity(address_city);
     address.setStateOrProvince(address_state);
     address.setPostalCode(address_zipcode);
     address.setPostalCodeExt(address_zip_ext);
     address.setCounty(address_county);
     address.setCountry(address_country);
  personBO.addAddress(person, add);
...
```

equals

Description

The **equals** method compares one Address object with another to see whether any fields are different. A null address field is considered to be different from a field that is empty but not null.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The Address object to be compared with the current Address object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The Address objects were compared successfully and all fields are identical between the two objects.

This value is returned …	if this occurs …
False	The Address objects were compared successfully and the fields are not identical between the two objects.

Throws

None.

Additional Information

Overrides:

- **equals** in the **java.lang.Object** class (see your Java documentation for more information about this method)

Example

To se an example of how **equals** in the Address class can be used, see the example for **addAddress** beginning on page 4-409.

getAddress1

Description

The **getAddress1** method retrieves the value of the **address1** field in an address record. You can use this method to display the first line of a person's address for an Address object.

Syntax

```
public java.lang.String getAddress1()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress1** method returns one of the following values:

This value is returned …	if this occurs …
String containing the first line in an address record	The **address1** field of the address record was retrieved successfully.
Null	The **address1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the *system_code*, *local_id*, and *status* variables). It then calls **loadPerson** to retrieve the information associated with that UID, and calls **getAddressEnumeration** to obtain a list of all addresses associated with the person record. The example calls **hasMoreElements** and **nextElement** to scroll through each address record. It then calls the '**get**' methods in the Address class to display the fields in each address record. Note that the **getAddressType** function retrieves the coded value of the address type, so **getDisplayValue** in the CodeLookup class is called to translate the coded value to the display value.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system,local_id,enum);
 personBO.loadPerson(uid, person);

 CodeLookup codeLookup = CodeLookup.getInstance(eiServer);
 Enumeration a = person.getAddressEnumeration();
 while (a.hasMoreElements()) {
     Address ad = (Address)a.nextElement();
     EnumCodeType codeType = EnumCodeType.ADDRESS_TYPE;
     String type = ad.getAddressType();
     String display = codeLookup.getDisplayValue(codeType, type);

     System.out.println(" Address Type: " + display);
     System.out.println(" Address 1: " + ad.getAddress1());
     System.out.println(" Address 2: " + ad.getAddress2());
     System.out.println(" Address 3: " + ad.getAddress3());
     System.out.println(" Address 4: " + ad.getAddress4());
     System.out.println(" City: " + ad.getCity());
     System.out.println(" State: " + ad.getStateOrProvince());
     System.out.println(" Zip: " + ad.getPostalCode());
     System.out.println(" Zip Extension: " + ad.getPostalCodeExt());
     System.out.println(" County: " + ad.getCounty());
     System.out.println(" Country: " + ad.getCountry());
     /*  Now display parsed address components */
     System.out.println(" Street Name: " + ad.getStreetName());
     System.out.println(" Street Direction: " + ad.getStreetDir());
     System.out.println(" House Number: " + ad.getHouseNumber());
     System.out.println(" Street Type: " + ad.getStreetType());

   }
...
```

getAddress2

Description

The **getAddress2** method retrieves the value of the **address2** field in an address record. You can use this method to display the second line of a person's address for an Address object.

Syntax

```
public java.lang.String getAddress2()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress2** method returns one of the following values:

This value is returned …	if this occurs …
String containing the second line in an address record	The **address2** field of the address record was retrieved successfully.
Null	The **address2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getAddress3

Description

The **getAddress3** method retrieves the value of the **address3** field in an address record. You can use this method to display the third line of a person's address for an Address object.

Syntax

```
public java.lang.String getAddress3()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress3** method returns one of the following values:

This value is returned …	if this occurs …
String containing the third line in an address record	The **address3** field of the address record was retrieved successfully.
Null	The **address3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getAddress4

Description

The **getAddress4** method retrieves the value of the **address4** field in an address record. You can use this method to display the fourth line of a person's address for an Address object.

Syntax

```
public java.lang.String getAddress4()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress4** method returns one of the following values:

This value is returned …	if this occurs …
String containing the fourth line in an address record	The **address4** field of the address record was retrieved successfully.
Null	The **address4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getAddressType

Description

The **getAddressType** method retrieves the value of the **addressType** field in an address record. You can use this method to display the type of address contained in an Address object.

Syntax

```
public java.lang.String getAddressType()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddressType** method returns the following value:

This value is returned …	if this occurs …
String containing an address type	The **addressType** field of the address record was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getCity

Description

The **getCity** method retrieves the value of the **city** field in an address record. You can use this method to display the city of a person's address for an Address object.

Syntax

```
public java.lang.String getCity()
```

Parameters

Parameter	Description
None	

Return Value

The **getCity** method returns one of the following values:

This value is returned …	if this occurs …
String containing a city name	The **city** field of the address record was retrieved successfully.
Null	The **city** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getCountry

Description

The **getCountry** method retrieves the value of the **country** field in an address record. You can use this method to display the country of a person's address for an Address object.

Syntax

```
public java.lang.String getCountry()
```

Parameters

Parameter	Description
None	

Return Value

The **getCountry** method returns one of the following values:

This value is returned …	if this occurs …
String containing a country code	The **country** field of the address record was retrieved successfully.
Null	The **country** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getCounty

Description

The **getCounty** method retrieves the value of the **county** field in an address record. You can use this method to display the county of a person's address for an Address object.

Syntax

```
public java.lang.String getCounty()
```

Parameters

Parameter	Description
None	

Return Value

The **getCounty** method returns one of the following values:

This value is returned …	if this occurs …
String containing a county	The **county** field of the address record was retrieved successfully.
Null	The **county** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getHouseNumber

Description

The **getHouseNumber** method retrieves the value of the **houseNumber** field in an address record. You can use this method to display the house number of a person's address for an Address object. The **houseNumber** field is the value of the house number as parsed from the street address by the Vality algorithm.

Syntax

```
public java.lang.String getHouseNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getHouseNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a number	The **houseNumber** field of the address record was retrieved successfully.
Null	The **houseNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getPostalCode

Description

The **getPostalCode** method retrieves the value of the **postalCode** field in an address record. You can use this method to display the postal code of a person's address for an Address object.

Syntax

```
public java.lang.String getPostalCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getPostalCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a postal code	The **postalCode** field of the address record was retrieved successfully.
Null	The **postalCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getPostalCodeExt

Description

The **getPostalCodeExt** method retrieves the value of the **postalCodeExt** field in an address record. You can use this method to display the postal code extension of a person's address for an Address object.

Syntax

```
public java.lang.String getPostalCodeExt()
```

Parameters

Parameter	Description
None	

Return Value

The **getPostalCodeExt** method returns one of the following values:

This value is returned …	if this occurs …
String containing a postal code extension	The **postalCodeExt** field of the address record was retrieved successfully.
Null	The **postalCodeExt** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getStateOrProvince

Description

The **getStateOrProvince** method retrieves the value of the **stateOrProvince** field in an address record. You can use this method to display the state or province of a person's address for an Address object.

Syntax

```
public java.lang.String getStateOrProvince()
```

Parameters

Parameter	Description
None	

Return Value

The **getStateOrProvince** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state or province code	The **stateOrProvince** field of the address record was retrieved successfully.
Null	The **stateOrProvince** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getStreetDir

Description

The **getStreetDir** method retrieves the value of the **streetDir** field in an address record. You can use this method to display the direction indicator for the street in a person's address for an Address object. The **streetDir** field is the value of the street direction as parsed from the street address by the Vality algorithm.

Syntax

```
public java.lang.String getStreetDir()
```

Parameters

Parameter	Description
None	

Return Value

The **getStreetDir** method returns one of the following values:

This value is returned …	if this occurs …
String containing a direction abbreviation	The **streetDir** field of the address record was retrieved successfully.
Null	The **streetDir** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getStreetName

Description

The **getStreetName** method retrieves the value of the **streetName** field in an address record. You can use this method to display the street name of a person's address for an Address object. The **streetName** field is the value of the street name as parsed from the street address by the Vality algorithm.

Syntax

```
public java.lang.String getStreetName()
```

Parameters

Parameter	Description
None	

Return Value

The **getStreetName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a street name	The **streetName** field of the address record was retrieved successfully.
Null	The **streetName** field was checked successfully but there was no value to retrieve.

Throws

The **getStreetName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

getStreetType

Description

The **getStreetType** method retrieves the value of the **streetType** field in an address record. You can use this method to display the type of street in a person's address for an Address object. The **streetType** field is the value of the street type as parsed from the street address by the Vality algorithm.

Syntax

```
public java.lang.String getStreetType()
```

Parameters

Parameter	Description
None	

Return Value

The **getStreetType** method returns one of the following values:

This value is returned …	if this occurs …
String containing a street type abbreviation	The **streetType** field of the address record was retrieved successfully.

This value is returned …	if this occurs …
Null	The **streetType** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an Address object, see the example for the **getAddress1** method on page 4-38.

setAddress1

Description

The **setAddress1** method is the setter for the **address1** field. Use this method to populate the street address of a person's address in an Address object.

Syntax

```
public void setAddress1(java.lang.String address1)
```

Parameters

Parameter	Description
address1	The value of the **address1** field for the Address object.

Return Value

None.

Throws

The **setAddress1** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setAddress2

Description

The **setAddress2** method is the setter for the **address2** field. Use this method to populate the second line of a person's address in an Address object.

Syntax

```
public void setAddress2(java.lang.String address2)
```

Parameters

Parameter	Description
address2	The value of the **address2** field for the Address object.

Return Value

None.

Throws

The **setAddress2** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setAddress3

Description

The **setAddress3** method is the setter for the **address3** field. Use this method to populate the third line of a person's address in an Address object.

Syntax

```
public void setAddress3(java.lang.String address3)
```

Parameters

Parameter	Description
address3	The value of the **address3** field for the Address object.

Return Value

None.

Throws

The **setAddress3** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setAddress4

Description

The **setAddress4** method is the setter for the **address4** field. Use this method to populate the street address of a person's address in an Address object.

Syntax

```
public void setAddress4(java.lang.String address4)
```

Parameters

Parameter	Description
address4	The value of the **address4** field for the Address object.

Return Value

None.

Throws

The **setAddress4** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setCity

Description

The **setCity** method is the setter for the **city** field. Use this method to populate the city of a person's address in an Address object.

Syntax

```
public void setCity(java.lang.String city)
```

Parameters

Parameter	Description
city	The value of the **city** field for the Address object.

Return Value

None.

Throws

The **setCity** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setCountry

Description

The **setCountry** method is the setter for the **setCountry** field. Use this method to populate the country of a person's address in an Address object.

Syntax

```
public void setCountry(java.lang.String country)
```

Parameters

Parameter	Description
country	The value of the **country** field for the Address object.

Return Value

None.

Throws

The **setCountry** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setCounty

Description

The **setCounty** method is the setter for the **county** field. Use this method to populate the county of a person's address in an Address object.

Syntax

```
public void setCounty(java.lang.String county)
```

Parameters

Parameter	Description
county	The value of the **county** field for the Address object.

Return Value

None.

Throws

The **setCounty** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setPostalCode

Description

The **setPostalCode** method is the setter for the **postalCode** field. Use this method to populate the postal code of a person's address in an Address object.

Syntax

```
public void setPostalCode(java.lang.String postalCode)
```

Parameters

Parameter	Description
postalCode	The value of the **postalCode** field for the Address object.

Return Value

None.

Throws

The **setPostalCode** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setPostalCodeExt

Description

The **setPostalCodeExt** method is the setter for the **postalCodeExt** field. Use this method to populate the postal code extension of a person's address in an Address object.

Syntax

```
public void setPostalCodeExt(java.lang.String postalCodeExt)
```

Parameters

Parameter	Description
postalCodeExt	The value of the **postalCodeExt** field for the Address object.

Return Value

None.

Throws

The **setPostalCodeExt** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

setStateOrProvince

Description

The **setStateOrProvince** method is the setter for the **stateOrProvince** field. Use this method to populate the state or province of a person's address in an Address object.

Syntax

```
public void setStateOrProvince(java.lang.String
stateOrProvince)
```

Parameters

Parameter	Description
stateOrProvince	The value of the **stateOrProvince** field for the Address object.

Return Value

None.

Throws

The **setStateOrProvince** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Address object, see the example for the **Address** method on page 4-35.

toString

Description

The **toString** method creates a string representation of the address record. You can use this method to display a comma-delimited list of address information.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing a complete address	The address was retrieved successfully.

Throws

None

Additional Information

Overrides:

- **toString** in the **com.stc.eIndex.active.core.DataObject** class

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the *system_code*, *local_id*, and *status* variables). It then calls **loadPerson** to retrieve the information associated with that UID, and calls **getAddressEnumeration** to obtain a list of all addresses associated with the person record. The example then calls **toString** to display the fields in each address record. The address fields are delimited by commas and each address record is surrounded by brackets, as shown in the example below.

```
[H,1330 BLOSSOM STREET,UNIT 5,null,null,CAPE BURR,CT,09876,
8007,CAPE BURR,UNST]
[O,3478 SHORELINE DRIVE,SUITE 1510,BILLING DEPARTMENT,null,
SHEFFIELD,CT,09877,null,CAPE BURR,UNST]
```

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system,local_id,enum);
 personBO.loadPerson(uid, person);

 CodeLookup codeLookup = CodeLookup.getInstance(eiServer);
 Enumeration a = person.getAddressEnumeration();
 while (a.hasMoreElements()) {
     Address ad = (Address)a.nextElement();
     System.out.println(ad.toString);
   }
...
```

Alias Class

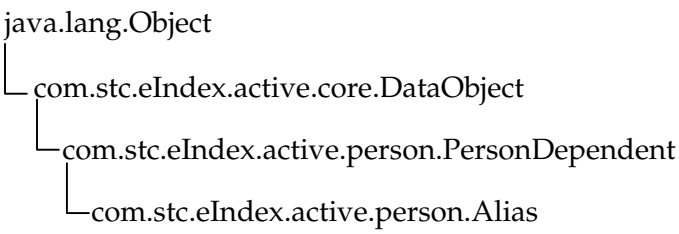
Description

The **Alias** class represents alias information associated with a person record. Use the methods in this class to create Alias objects and to populate or retrieve fields in an Alias object. You can also display a comma-delimited string of alias information using the **toString** method.

Properties

The **Alias** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**



Constructor

The **Alias** class has one constructor, which is described on the following page:

- **Alias** on page 4-60

Methods

The methods included in the **Alias** class are described in detail on the following pages:

- **equals** on page 4-61
- **getFirstName** on page 4-62
- **getLastName** on page 4-63
- **getMiddleName** on page 4-64
- **getUid** on page 4-65
- **setFirstName** on page 4-65
- **setLastName** on page 4-67
- **setMiddleName** on page 4-67
- **toString** on page 4-68

Inherited Methods

The **Alias** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Alias

Description

The **Alias** method is the constructor method for the Alias class. Use this method to create objects containing a person's alias information.

Syntax

```
public Alias(java.lang.String firstName, java.lang.String
lastName, java.lang.String middleName)
```

Parameters

Parameter	Description
firstName	The first name of the alias name.
lastName	The last name of the alias name.
middleName	The middle name or initial of the alias name.

Return Value

The **Alias** constructor method returns the following value:

This value is returned ...	if this occurs ...
An object containing alias information	The Alias object was created successfully.

Throws

The **Alias** constructor throws the following exception:

- **EiException Class**

Example

The following example creates a new Alias object with first name **SAMUEL**, last name **WARREN**, and no middle name. It then calls **hasAlias** to see if the specified Person object is already associated with the new alias information. If the Person object is not associated with the alias, **addAlias** is called to add the new alias information to the Person object.

```
...
  Alias a = new Alias("SAMUEL","WARREN","");
  if (!personBO.hasAlias(person,a)) {
      personBO.addAlias(person, a);
...
```

equals

Description

The **equals** method compares one Alias object with another to see whether any fields are different. This comparison is case-insensitive.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The Alias object to be compared with the current Alias object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The Alias objects were compared successfully and all fields are identical between the two objects.
False	The Alias objects were compared successfully and the fields are not identical between the two objects.

Throws

None.

Additional Information

Overrides:

- **equals** in the **java.lang.Object** class (see your Java documentation for more information about this method)

Example

To see an example of how **equals** can be used to compare two Alias objects, see the example for **setFirstName** on page 4-65.

getFirstName

Description

The **getFirstName** method retrieves the value of the **firstName** field in an alias record. You can use this method to display the first name for an Alias object.

Syntax

```
public java.lang.String getFirstName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFirstName** method returns the following value:

This value is returned …	if this occurs …
String containing the first name of an alias name	The **firstName** field of the alias record was retrieved successfully.

Throws

None.

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getAliasEnumeration** to obtain a list of all alias names associated with the person record. The sample retrieves the individual fields from each alias record by calling the '**get**' methods in the Alias class.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration a = person.getAliasEnumeration();
 while (a.hasMoreElements()) {
     Alias an = (Alias)a.nextElement();
     System.out.println(" First Name: " + an.getFirstName());
     System.out.println(" Last Name: " + an.getLastName());
     System.out.println(" Middle Name: " + an.getMiddleName());
     System.out.println(" UID: " + an.getUid());
 }
...
```

getLastName

Description

The **getLastName** method retrieves the value of the **lastName** field in an alias record. You can use this method to display the last name for an Alias object.

Syntax

```
public java.lang.String getLastName()
```

Parameters

Parameter	Description
None	

Return Value

The **getLastName** method returns the following value:

This value is returned …	if this occurs …
String containing the last name of an alias name	The **lastName** field of the alias record was retrieved successfully.

Throws

None.

Example

To see an example of how **getLastName** can be used, see the example for **getFirstName** on page 4-62.

getMiddleName

Description

The **getMiddleName** method retrieves the value of the **middleName** field in an alias record. You can use this method to display the middle name or initial for an Alias object.

Syntax

```
public java.lang.String getMiddleName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMiddleName** method returns one of the following values:

This value is returned …	if this occurs …
String containing the middle name or initial of an alias name	The **middleName** field of the alias record was retrieved successfully.
Null	The **middleName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getMiddleName** can be used, see the example for **getFirstName** on page 4-62.

getUid

Description

The **getUid** method retrieves the UID associated with the alias record. You can use this method to display the UID associated with an Alias object.

Syntax

```
public Uid getUid()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid** method returns the following value:

This value is returned …	if this occurs …
Uid object	The UID associated with the alias record was retrieved successfully.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

To see an example of how **getUid** can be used, see the example for **getFirstName** on page 4-62.

setFirstName

Description

The **setFirstName** method is the setter for the **firstName** field. Use this method to update the first name of a person's alias in an Alias object.

Syntax

```
public void setFirstName(java.lang.String firstName)
```

Parameters

Parameter	Description
firstName	The value of the **firstName** field for the Alias object.

Return Value

None.

Throws

The **setFirstName** method throws the following exception:

- **EiException Class**

Example

The following example illustrates how the '**set**' methods in the Alias class can be used when updating a Person object. The example creates a new Person object, and then searches for the UID of the person to update (based on the user-defined variables *system_code*, *local_id*, and *status*). The example calls **loadPerson** to load the Person object. It creates a new Alias object, **alias**, that defines the alias record to be updated. The example then retrieves an enumeration of Alias objects from the Person object and scrolls through the alias records searching for one that matches the previously defined **alias** object. When a matching alias record is found, it is updated using the '**set**' methods in the Alias class.

```
...
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
 personBO.loadPerson(uid, person);

 Alias alias = new Alias("BETH", "COLTER", "");
 Enumeration n = person.getAliasEnumeration();
 while (n.hasMoreElements()) {
     Alias an = (Alias)n.nextElement();
     if (an.equals(alias)) {
         an.setFirstName("LIZBETH");
         an.setLastName("COLTER");
         an.setMiddleName("J"); }
 }

 .../* Creating and populating Transaction object trans */
 boolean updatePerformed = personBO.updatePerson(person, trans);
 if (updatePerformed) {
     System.out.println("Update complete."); }
 else {
     System.out.println("Record unchanged."); }
...
```

setLastName

Description

The **setLastName** method is the setter for the **lastName** field. Use this method to update the last name of a person's alias in an Alias item.

Syntax

```
public void setLastName(java.lang.String lastName)
```

Parameters

Parameter	Description
lastName	The value of the **lastName** field for the Alias object.

Return Value

None.

Throws

The **setLastName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Alias object, see the example for **setFirstName** beginning on page 4-66.

setMiddleName

Description

The **setMiddleName** method is the setter for the **setMiddleName** field. Use this method to update the middle name or initial of a person's alias in an Alias object.

Syntax

```
public void setMiddleName(java.lang.String middleName)
```

Parameters

Parameter	Description
middleName	The value of the **middleName** field for the Alias object.

Return Value

None.

Throws

The **setMiddleName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Alias object, see the example for **setFirstName** beginning on page 4-66.

toString

Description

The **toString** method creates a string representation of the alias record. You can use this method to display a comma-delimited list of alias information.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing a complete alias name	The alias name was retrieved successfully.

Throws

None

Additional Information

Overrides:

- **toString** in the **com.stc.eIndex.active.core.DataObject** class

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information

associated with that UID, and calls **getAliasEnumeration** to obtain a list of all alias names associated with the person record.

The sample retrieves each alias record by calling **toString** in the Alias class. The elements in each alias record are delimited by commas, and each record is surrounded by brackets, as shown in this example list.

```
[ELIZABETH,WARREN,null]
[LIZ,SMITH,null]
[LIZ,WARREN,null]
[ELIZABETH,SMITH,J]
```

```
...
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration a = person.getAliasEnumeration();
 while (a.hasMoreElements()) {
     Alias an = (Alias)a.nextElement();
     System.out.println(an.toString());
...
```

Audit Class

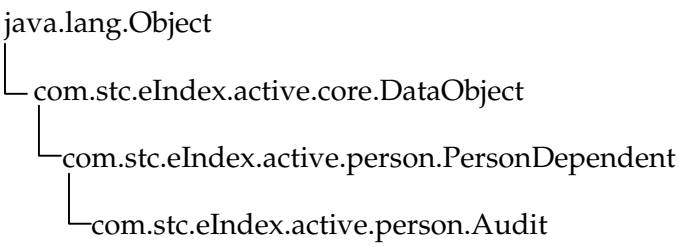
Description

The **Audit** class represents information stored in the *ui_audit* table. Use the methods in the **Audit** class to store and look up specific instances of user access of person information in the e*Index database.

Properties

The **Audit** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**



Constructor

The **Audit** class has one constructor, which is described on the following page:

- **Audit** on page 4-72

Methods

The methods included in the **Audit** class are described in detail on the following pages:

- **getDetail** on page 4-73
- **getFunct** on page 4-74
- **getTimestamp** on page 4-74
- **getUid1** on page 4-75
- **getUid2** on page 4-76
- **getUserId** on page 4-76
- **setDetail** on page 4-77
- **setFunct** on page 4-78
- **setTimestamp** on page 4-78
- **setUid2** on page 4-79
- **setUserId** on page 4-80

Inherited Methods

The **Audit** class inherits these methods from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **Audit** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Audit

Description

The **Audit** method is the constructor method for the Audit class. Use this method to create objects containing audit log information. The audit log stores information about each instance the *ui_person* table is accessed.

Syntax

```
public Audit()
```

Parameters

Parameter	Description
None	

Return Value

The **Audit** constructor method returns the following value:

This value is returned …	if this occurs …
An object containing audit information	The Audit object was created successfully.

Throws

None.

Example

The following example is excerpted from a sample used to create a new Person object and then create an audit log record for the transaction. It obtains a new instance of EiServer and PersonBO, and then creates an empty Person object. The example creates a Demographics object to populate the demographic fields of the Person object., and creates an Audit object to create the audit log record. The '**set**' methods are called to fill the Audit object with audit information, and **addAudit** is called to add the Audit object to the Person object.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();

 Uid uid = new Uid("1000000001");
 personBO.loadPerson(uid, person);
 .../* Updating the person object using setter methods */

 boolean updatePerformed = personBO.updatePerson(person, trans);
    if (updatePerformed)
      Audit audit = new Audit();
        audit.setUserId(user_id);
        audit.setFunct(function);
        audit.setDetail(function_description);
        audit.setTimestamp(eiServer.getDbTimestamp());
      personBO.addAudit(person, audit);
      System.out.println("Update complete.");
...
```

getDetail

Description

The **getDetail** method retrieves the value of the **detail** field in an audit record. You can use this method to display a description of the transaction that caused the audit record to be written.

Syntax

```
public java.lang.String getDetail()
```

Parameters

Parameter	Description
None	

Return Value

The **getDetail** method returns one of the following values:

This value is returned …	if this occurs …
String containing audit information	The **detail** field was retrieved successfully.
Null	The **detail** field was checked successfully but there was no value to retrieve.

Throws

None.

getFunct

Description

The **getFunct** method retrieves the value of the **funct** field in an Audit object. You can use this method to display the event code that caused an audit log entry to be written.

Syntax

```
public java.lang.String getFunct()
```

Parameters

Parameter	Description
None	

Return Value

The **getFunct** method returns one of the following values:

This value is returned …	if this occurs …
String containing an event type code information	The **funct** field was retrieved successfully.
Null	The **funct** field was checked successfully but there was no value to retrieve.

Throws

None.

getTimestamp

Description

The **getTimestamp** method retrieves the value of the **timestamp** field in an Audit object. You can use this method to display the date and time an audit log entry was created.

Syntax

```
public java.sql.Timestamp getTimestamp()
```

Parameters

Parameter	Description
None	

Return Value

The **getTimestamp** method returns the following value:

This value is returned …	if this occurs …
A timestamp	The **timestamp** field was retrieved successfully.

Throws

The **getTimestamp** method throws the following exception:

- **EiException Class**

getUid1

Description

The **getUid1** method retrieves the value of the **uid1** field for the first person record accessed in the specified transaction. You can use this method to display the value of the UID associated with the audit log entry.

Syntax

```
public Uid getUid1()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid1** method returns one of the following values:

This value is returned …	if this occurs …
A Uid object	The **uid1** field was retrieved successfully.
Null	The **uid1** field was checked successfully, but there was no value to retrieve.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-493.

getUid2

Description

The **getUid2** method retrieves the value of the **uid2** field for the second person record accessed in the specified transaction. You can use this method to display the value of the **uid2** field.

Syntax

```
public Uid getUid2()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid2** method returns the following value:

This value is returned …	if this occurs …
A Uid object	The **uid2** field was retrieved successfully.
Null	The **uid2** field was checked successfully but there was no value to retrieve.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-493.

getUserId

Description

The **getUserId** method retrieves the value of the **userId** field in the specified transaction. You can use this method to display the user ID of the user who performed the transaction that caused an audit log record to be logged.

Syntax

```
public java.lang.String getUserId()
```

Parameters

Parameter	Description
None	

Return Value

The **getUserId** method returns the following value:

This value is returned …	if this occurs …
String containing a user ID	The **userId** field was retrieved successfully.

Throws

None.

setDetail

Description

The **setDetail** method is the setter for the audit **detail** field. Use this method to populate the detail information in an Audit object.

Syntax

```
public void setDetail(java.lang.String detail)
```

Parameters

Parameter	Description
detail	The value of the **detail** field for the new Audit object.

Return Value

None.

Throws

The **setDetail** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Audit object, see the example for **Audit** on page 4-72.

setFunct

Description

The **setFunct** method is the setter for the audit **funct** field. Use this method to populate the event type of a transaction that creates an Audit object.

Syntax

```
public void setFunct(java.lang.String funct)
```

Parameters

Parameter	Description
funct	The value of the **funct** field for the new Audit object.

Return Value

None.

Throws

The **setFunct** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Audit object, see the example for **Audit** on page 4-72.

setTimestamp

Description

The **setTimestamp** method is the setter for the audit **timestamp** field. Use this method to populate the date a transaction occurred in an Audit object.

Syntax

```
public void setTimestamp(java.sql.Timestamp date)
```

Parameters

Parameter	Description
date	The value of the **timestamp** field for the new Audit object.

Return Value

None.

Throws

The **setTimestamp** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Audit object, see the example for **Audit** on page 4-72.

setUid2

Description

The **setUid2** method is the setter for the audit **uid2** field for the second person record accessed during a transaction. Use this method to populate the UID of the second person record in an Audit object. The primary UID is automatically populated during a transaction.

Syntax

```
public void setUid2(Uid uid2)
```

Parameters

Parameter	Description
uid2	The value of the **uid2** field for the new Audit object.

Return Value

None.

Throws

The **setUid2** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

To see an example of how the '**set**' methods can be used to populate an Audit object, see the example for **Audit** on page 4-72.

setUserId

Description

The **setUserId** method is the setter for the audit **userId** field. Use this method to populate the user ID of the person who performed the transaction recorded in an Audit object.

Syntax

```
public void setUserId(java.lang.String userId)
```

Parameters

Parameter	Description
userId	The value of the **userId** field for the new Audit object.

Return Value

None.

Throws

The **setUserId** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate an Audit object, see the example for **Audit** on page 4-72.

AuxId Class

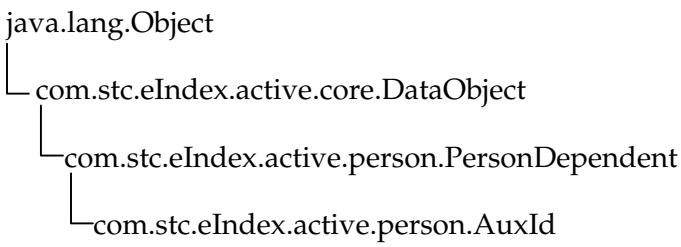
Description

The **AuxId** class represents non-unique ID information (also known as *auxiliary IDs*) for the Person class. Use the methods in this class to retrieve and update non-unique ID information in a person record.

Properties

The **AuxId** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**



Constructor

The **AuxId** class has one constructor, which is described on the following page:

- **AuxId** on page 4-83

Methods

The methods included in the **AuxId** class are described in detail on the following pages:

- **equals** on page 4-84
- **getAuxIdType** on page 4-84
- **getId** on page 4-86
- **setAuxIdType** on page 4-86
- **setId** on page 4-88
- **toString** on page 4-88

Inherited Methods

The **AuxId** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

AuxId

Description

The **AuxId** method is the constructor method for the AuxId class. Use this method to create objects containing a person's non-unique ID information.

Syntax

```
public AuxId(java.lang.String idType, java.lang.String id)
```

Parameters

Parameter	Description
idType	The non-unique ID code of the ID. This value must already be defined in the *ui_aux_id_def* table and must be less than eight characters.
id	The identification code for the new AuxId object.

Return Value

The **AuxId** constructor method returns the following value:

This value is returned ...	if this occurs ...
An object having a non-unique ID and type	The AuxId object was created successfully.

Throws

The **AuxId** method throws the following exceptions:

- **EiException Class**

Example

The following example obtains a new instance of EiServer and PersonBO, and then creates an empty Person object. It creates a Demographics object to populate the demographic fields of the Person object, and a new AuxId object, **auxId**, to add the non-unique ID information to the person record.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 Demographics demo = new Demographics();
     demo.setLastName(last_name);
     demo.../* Populating property values */
 ...
 AuxId auxId = new AuxId("ACCT", "100495");
 personBO.addAuxId(person, auxId);
 ...
```

equals

Description

The **equals** method compares one AuxId object with another to see whether any fields are different.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The AuxId object to be compared with the current AuxId object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The AuxId objects were compared successfully and all fields are identical between the two objects.
False	The AuxId objects were compared successfully and the fields are not identical between the two objects.

Throws

None.

Additional Information

Overrides:

- **equals** in the **java.lang.Object** class (see your Java documentation for more information about this method)

Example

To see an example of how the **equals** method can be used to compare two AuxId objects, see the example for **setAuxIdType** on page 4-86.

getAuxIdType

Description

The **getAuxIdType** method returns the type of non-unique ID in an AuxId object.

Syntax

```
public java.lang.String getAuxIdType()
```

Parameters

Parameter	Description
None	

Return Value

The **getAuxIdType** method returns the following value:

This value is returned …	if this occurs …
String containing a non-unique ID type	The **auxIdType** field was successfully retrieved.

Throws

None.

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the *system_code*, *local_id*, and *status* variables). It then calls **loadPerson** to retrieve the information associated with that UID, and calls **getAuxIdEnumeration** to obtain a list of all non-unique IDs associated with the person record. The example calls **hasMoreElements** and **nextElement** to scroll through each ID record. It then calls the '**get**' methods in the AuxId class to display the fields in each ID record. **How to display type description instead of type code?**

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration x = person.getAuxIdEnumeration();
 while (x.hasMoreElements()) {
    AuxId aux = (AuxId)x.nextElement();
    System.out.println(" Non-unique ID Type: " + aux.getAuxIdType());
    System.out.println(" ID: " + aux.getId());
 }
...
```

getId

Description

The **getId** method returns the identification code associated with the non-unique ID type in an AuxId object.

Syntax

```
public java.lang.String getId()
```

Parameters

Parameter	Description
None	

Return Value

The **getId** method returns the following value:

This value is returned …	if this occurs …
String containing an identification code	The **id** field was successfully retrieved.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from an AuxId object, see the example for the **getAuxIdType** method on page 4-85.

setAuxIdType

Description

The **setAuxType** method is the setter for the **idType** field. Use this method to update the non-unique ID type in an AuxId object.

Syntax

```
public void setAuxIdType(java.lang.String idType)
```

Parameters

Parameter	Description
idType	The value of the **idType** field for the new AuxId object.

Return Value

None.

Throws

The **setAuxIdType** method throws the following exception:

- **EiException Class**

Example

The following example illustrates how the '**set**' methods in the AuxId class can be used when updating a Person object. The example creates a new Person object, and then searches for the UID of the person to update (based on the user-defined variables *system_code*, *local_id*, and *status*). The example calls **loadPerson** to load the Person object. It creates a new AuxId object, **aux**, which defines the non-unique ID record to be updated. The example then retrieves an enumeration of AuxId objects from the Person object and scrolls through the ID records searching for one that matches the previously defined **aux** object. When a matching ID record is found, it is updated using the '**set**' methods in the AuxId class.

```
...
Person person = new Person();
EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);
personBO.loadPerson(uid, person);

AuxId aux = new AuxId("ACCT", "03257293");
Enumeration enum = person.getAuxIdEnumeration();
while (enum.hasMoreElements()) {
    AuxId a = (AuxId)enum.nextElement();
    if (a.equals(aux)) {
        a.setAuxIdType("ACCT");
        a.setId("03257392");
    }
}

.../* Creating and populating Transaction object trans */
boolean updatePerformed = personBO.updatePerson(person, trans);
if (updatePerformed) {
    System.out.println("Update complete.");
} else {
    System.out.println("Record unchanged.");
}
...
```

setId

Description

The **setId** method is the setter for the **id** field. Use this method to update the non-unique identification code in an AuxId object.

Syntax

```
public void setId(java.lang.String id)
```

Parameters

Parameter	Description
id	The value of the **id** field for the new AuxId object.

Return Value

None.

Throws

The **setId** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to update an AuxId object, see the example for **setAuxIdType** on page 4-86.

toString

Description

The **toString** method retrieves a comma-delimited representation of an AuxId object.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns one of the following values:

This value is returned …	if this occurs …
Comma-delimited list of AuxId data	The specified non-unique IDs for the record were retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in the **com.stc.eIndex.active.core.DataObject** class

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the *system_code*, *local_id*, and *status* variables). It then calls **loadPerson** to retrieve the information associated with that UID, and calls **getAuxIdEnumeration** to obtain a list of all non-unique IDs associated with the person record. The example scrolls through each ID record using the **nextElement** and **hasMoreElements** functions. It then calls the **toString** method in the AuxId class to display string representations of each non-unique ID.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration x = person.getAuxIdEnumeration();
 while (x.hasMoreElements()) {
    AuxId aux = (AuxId)x.nextElement();
    System.out.println(aux.toString());
 }
...
```

CityState Class

Description

The **CityState** class is used by the zip code lookup function to return a city, state, county, and residence code information for a ZipCodeLookup object.

Properties

The **CityState** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└ com.stc.eIndex.active.person.CityState

Constructor

None.

Methods

The methods included in the **CityState** class are described in detail on the following pages:

- **getCity** on page 4-91
- **getCounty** on page 4-91
- **getResidenceCode** on page 4-92
- **getState** on page 4-93

Inherited Methods

The **CityState** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCity

Description

The **getCity** method retrieves the city associated with the specified zip code.

Syntax

```
public java.lang.String getCity()
```

Parameters

Parameter	Description
None	

Return Value

The **getCity** method returns one of the following values:

This value is returned …	if this occurs …
String containing a city name	The value of the **city** field was successfully retrieved.
Null	No city was associated with the specified zip code.

Throws

None.

Example

To see an example of how **getCity** in the CityState class can be used, see the example code provided for **getCityState** on page 4-498.

getCounty

Description

The **getCounty** method retrieves the county associated with the specified zip code.

Syntax

```
public java.lang.String getCounty()
```

Parameters

Parameter	Description
None	

Return Value

The **getCounty** method returns one of the following values:

This value is returned …	if this occurs …
String containing a county code	The value of the **county** field was successfully retrieved.
Null	No **county** was associated with the specified zip code.

Throws

None.

Example

To see an example of how **getCounty** in the CityState class can be used, see the example code provided for **getCityState** on page 4-498.

getResidenceCode

Description

The **getResidenceCode** method retrieves the residence code associated with the specified zip code.

Syntax

```
public java.lang.String getResidenceCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getResidenceCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a residence code	The value of the **residenceCode** field was successfully retrieved.
Null	No residence code was associated with the specified zip code.

Throws

None.

Example

To see an example of how **getResidenceCode** in the CityState class can be used, see the example code provided for **getCityState** on page 4-498.

getState

Description

The **getState** method retrieves the state associated with the specified zip code.

Syntax

```
public java.lang.String getState()
```

Parameters

Parameter	Description
None	

Return Value

The **getState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **state** field was successfully retrieved.
Null	No state was associated with the specified zip code.

Throws

None.

Example

To see an example of how **getState** in the CityState class can be used, see the example code provided for **getCityState** on page 4-498.

CodeDisplay Class

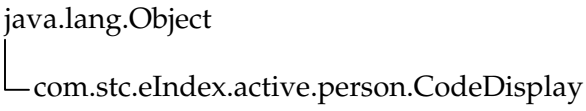
Description

The **CodeDisplay** class represents a processing code record returned by the **CodeLookup** class. Processing code records are stored in the e*Index database tables *stc_common_detail* and *stc_common_header*.

Properties

The **CodeDisplay** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Methods

The methods included in the **CodeDisplay** class are described in detail on the following pages:

- **getCode** on page 4-95
- **getCommonDetailId** on page 4-95
- **getCommonHeaderId** on page 4-96
- **getCreateDate** on page 4-97
- **getCreateUserId** on page 4-97
- **getDisplay** on page 4-98
- **getReadOnly** on page 4-99

Inherited Methods

The **CodeDisplay** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCode

Description

The **getCode** method retrieves the code value of the code table record represented by the **CodeLookup** class.

Syntax

```
public java.lang.String getCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a coded value	The value of the **code** field was retrieved successfully.
Null	There was no value in the **code** field to retrieve.

Throws

None.

Example

For an example of how **getCode** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getCommonDetailId

Description

The **getCommonDetailId** method retrieves the common detail ID of the code table record represented by the **CodeLookup** class. You only need to retrieve this value for code table records that are stored in the *stc_common_detail* table.

Syntax

```
public long getCommonDetailId()
```

Parameters

Parameter	Description
None	

Return Value

The **getCommonDetailId** method returns one of the following values:

This value is returned …	if this occurs …
Long integer	The value of the **commonDetailId** field was retrieved successfully.
Null	There was no value in the **commonDetailId** field to retrieve.

Throws

None.

Example

For an example of how **getCommonDetailId** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getCommonHeaderId

Description

The **getCommonHeaderId** method retrieves the common header ID of the code table record represented by the **CodeLookup** class.

Syntax

```
public long getCommonDetailId()
```

Parameters

Parameter	Description
None	

Return Value

The **getCommonHeaderId** method returns one of the following values:

This value is returned …	if this occurs …
Long integer	The value of the **commonHeaderId** field was retrieved successfully.
Null	There was no value in the **commonHeaderId** field to retrieve.

Throws

None.

Example

For an example of how **getCommonHeaderId** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getCreateDate

Description

The **getCreateDate** method retrieves the date the code table record represented by the **CodeLookup** class was created.

Syntax

```
public jave.sql.Timestamp getCreateDate()
```

Parameters

Parameter	Description
None	

Return Value

The **getCreateDate** method returns the following value:

This value is returned …	if this occurs …
A timestamp	The create date for the record was retrieved successfully.

Throws

None.

Example

For an example of how **getCreateDate** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getCreateUserId

Description

The **getCreateUserId** method retrieves the ID of the user who created the code table record represented by the **CodeLookup** class.

Syntax

```
public java.lang.String getCreateUserId()
```

Parameters

Parameter	Description
None	

Return Value

The **getCreateUserId** method returns one of the following values:

This value is returned …	if this occurs …
String containing a user ID	The value of the **userId** field was retrieved successfully.
Null	There was no value in the **userId** field to retrieve.

Throws

None.

Example

For an example of how **getCreateUserId** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getDisplay

Description

The **getDisplay** method retrieves the display value of the processing code record represented by the **CodeLookup** class.

Syntax

```
public java.lang.String getDisplay()
```

Parameters

Parameter	Description
None	

Return Value

The **getDisplay** method returns one of the following values:

This value is returned …	if this occurs …
String containing a display value	The value of the **display** field was retrieved successfully.
Null	There was no value in the **display** field to retrieve.

Throws

None.

Example

For an example of how **getDisplay** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

getReadOnly

Description

The **getReadOnly** method retrieves the read-only flag for the code table record represented by the **CodeLookup** class. Use this method to determine whether a record in *stc_common_detail* can be modified.

Syntax

```
public java.lang.String getReadOnly()
```

Parameters

Parameter	Description
None	

Return Value

The **getReadOnly** method returns one of the following values:

This value is returned …	if this occurs …
String containing a read-only flag	The value of the **readOnly** field was retrieved successfully.

Throws

None.

Example

For an example of how **getReadOnly** can be used, see the example for **getCodeDisplayPairs** on page 4-101.

CodeLookup Class

Description

The **CodeLookup** class looks up the display values and processing codes for the given processing code type.

Properties

The **CodeLookup** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└com.stc.eIndex.active.person.CodeLookup

Constructor

None.

Methods

The methods included in the **CodeLookup** class are described in detail on the following pages:

- **getCodeDisplayPairs** on page 4-101
- **getDisplayValue** on page 4-102
- **getInstance** on page 4-104

Inherited Methods

The **CodeLookup** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCodeDisplayPairs

Description

The **getCodeDisplayPairs** method returns a listing of all the CodeDisplay pairs for a given type of processing code.

Syntax

```
public java.util.Enumeration getCodeDisplayPairs(EnumCodeType codeType)
```

Parameters

Parameter	Description
codeType	The type of processing code for which you want to display codes and their associated descriptions. For a list of possible code types, see the fields listed for the **EnumCodeType Class** on page 4-352.

Return Value

The **getCodeDisplayPairs** method returns on of the following value:

This value is returned …	if this occurs …
Enumeration of coded values	An enumeration of the values of the CodeDisplay class was created successfully.
Null	There were no values defined for the specified code type.

Throws

The **getCodeDisplayPairs** throws the following exception:

- **java.sql.SQLException**

Example

The following example looks up the EnumCodeType for the specified type of processing code, represented by *proc_code*. The example initializes the variable **ect** and then retrieves an instance of EiServer. It then defines the variable **type**, and checks the type value against the possible options. It calls **EnumCodeType** to retrieve the enumeration for the specified code type.

Once the EnumCodeType is retrieved, **getInstance** is called to retrieve the instance of the CodeLookup class and **getCodeDisplayPairs** is called to retrieve the list of codes and associated display values for the given code type. The results are contained in the enumeration **ep**. The '**get**' methods in the CodeDisplay class are called to display the resulting information for codes and display values.

```
...
  EnumCodeType ect = null;
  EiServer eiServer = new EiServer("EiServer.properties");

    String type = proc_code;

    if (type.equals("sex"))
      ect = EnumCodeType.GENDER;
    else if (type.equals("mstatus"))
      ect = EnumCodeType.MARITAL_STATUS;
    else if (type.equals("title"))
      ect = EnumCodeType.TITLE;
    else if (type.equals("suffix"))
      ect = EnumCodeType.SUFFIX;
    ...

    if (ect != null) {
      CodeLookup codeLookup = CodeLookup.getInstance(eiServer);
      Enumeration ep = codeLookup.getCodeDisplayPairs(ect);
      System.out.println("  Values for code type: " + type);
      while (ep.hasMoreElements()) {
        CodeDisplay cd = (CodeDisplay)ep.nextElement();
        System.out.println("Code: " + cd.getCode());
        System.out.println("Display: " + cd.getDisplay());
        System.out.println("Read Only?  " cd.getReadOnly());
        System.out.println("Common Header ID:  " cd.getCommonHeaderId());
        System.out.println("Common Detail ID:  " cd.getCommonDetailId());
        System.out.println("Create Date:  " cd.getCreateDate());
        System.out.println("User ID: "  + cd.getCreateUserId());
     }
    }
  }
...
```

getDisplayValue

Description

The **getDisplayValue** method returns a display value given the value of the processing code.

Syntax

```
public java.lang.String getDisplayValue(EnumCodeType codeType,
java.lang.String code)
```

Parameters

Parameter	Description
codeType	The type of processing code for which you want to display codes and their associated descriptions. For a list of possible code types, see the fields listed for the **EnumCodeType Class** beginning on page 4-356.
code	The processing code associated with the display value you want to retrieve.

Return Value

The **getDisplayValue** method returns one of these values:

This value is returned …	if this occurs …
String containing a display value	The display value was retrieved successfully.
Null	There was no display value to retrieve.

Throws

The **getDisplayValue** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Example

The example below displays the display value associated with the specified processing code type and processing code. The example initializes the variable **ect**, retrieves an instance of EiServer, and then defines the variables **type** and **code** as the user-defined variables *code_type* and *proc_code*. It then checks the value of **type** against the possible options and calls **EnumCodeType** to retrieve the enumeration of the type.

The example then calls **getInstance** in the CodeLookup class, and calls **getDisplayValue** to retrieve the display description associated with the code and code type specified. Finally, the example displays the given processing code and its display value.

```
...
EnumCodeType ect = null;
  EiServer eiServer = new EiServer("EiServer.properties");

    String type = code_type;
    String code = proc_code;

    if (type.equals("sex"))
      ect = EnumCodeType.GENDER;
    else if (type.equals("mstatus"))
      ect = EnumCodeType.MARITAL_STATUS;
    else if (type.equals("title"))
      ect = EnumCodeType.TITLE;
    else if (type.equals("suffix"))
      ect = EnumCodeType.SUFFIX;
    ...

    if (ect != null) {
      CodeLookup codeLookup = CodeLookup.getInstance(eiServer);
      String display = codeLookup.getDisplayValue(ect, code);
      System.out.println("Code:" + code + "  Display:" + display);

  } else {
      System.out.println("  Required Parameter Missing");
  }
...
```

getInstance

Description

The **getInstance** method retrieves the single instance of the CodeLookup class for the given EiServer object, maintaining a singleton pattern so database results are stored for servicing subsequent lookups.

Syntax

```
public static CodeLookup getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the CodeLookup instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of CodeLookup	A CodeLookup instance for the specified EiServer object was retrieved successfully.

Throws

The **getInstance** method throws the following exceptions:

- **java.sql.SQLException**

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

For an example of how **getInstance** in the CodeLookup class can be used, see the example for **getCodeDisplayPairs** on page 4-101.

ControlKey Class

Description

The **ControlKey** class provides access to the control key information as defined in e*Index Administrator. This information is stored in the database table *ui_control*.

Properties

The **ControlKey** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└ com.stc.eIndex.active.person.ControlKey

Constructor

None.

Methods

The method included in the **ControlKey** class is described in detail on the following page:

- **isBlankOnUpdateEnabled** on page 4-125
- **isDobYearRequired** on page 4-126
- **isDuplicateCheckEnabled** on page 4-127
- **isExtensiveSearchEnabled** on page 4-128
- **isMixedCaseEnabled** on page 4-129
- **isOneExactMatchEnabled** on page 4-130
- **isSameFacilityReport** on page 4-131
- **isSeeDeactivatedEnabled** on page 4-132
- **isSeeMergedEnabled** on page 4-133
- **isShortIdEnabled** on page 4-134

Inherited Methods

The **ControlKey** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

For More Information

Other SeeBeyond publications may help you to learn how the control keys provided with e*Index determine processing rules.

To learn more about …	See …
Control Keys	Chapter 5 of the *e*Index Administrator User's Guide*
How control keys affect the standard e*Index e*Ways	Chapter 4 of the *e*Index Global Identifier Technical Reference*

getCheckSumLength

Description

The **getCheckSumLength** method retrieves the value of the CKSUMLEN control key, which controls the length of the check sum value for the UIDs created by e*Index.

Syntax

```
public int getCheckSumLength()
```

Parameters

Parameter	Description
None	

Return Value

The **getCheckSumLength** method returns one of the following values:

This value is returned …	if this occurs …
An integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the CKSUMLEN control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getCheckSumLength** to retrieve the value for the CKSUMLEN control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
String checksum = contrlKey.getCheckSumLength();
System.out.println("Checksum Length: " + checksum);
```

getCountry

Description

The **getCountry** method retrieves the value of the COUNTRY control key, which controls country-specific attributes of e*Index.

Syntax

```
public java.lang.String getCountry()
```

Parameters

Parameter	Description
None	

Return Value

The **getCountry** method returns one of the following values:

This value is returned …	if this occurs …
String containing a control key value	The control key was checked successfully and its value was returned.
Null	No value was associated with the COUNTRY control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getCountry** to retrieve the value for the COUNTRY control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
String country = contrlKey.getCounty();
System.out.println("Country: " + country);
```

getDateFormat

Description

The **getDateFormat** method retrieves the value of the DATEFRMT control key. This control key determines the format in which dates are displayed (for example, MMDDYYYY, DDMMYYYY, and so on).

Syntax

```
public java.lang.String getDateFormat()
```

Parameters

Parameter	Description
None	

Return Value

The **getDateFormat** method returns one of the following values:

This value is returned …	if this occurs …
String containing a control key value	The control key was checked successfully and its value was returned.
Null	No value was associated with the DATEFRMT control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getDateFormat** to retrieve the value for the DATEFRMT control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
String dateFormat = contrlKey.getDateFormat()
System.out.println("Date format: " + dateFormat);
```

getDobSearchRange

Description

The **getDobSearchRange** method retrieves the value of the SRCHDOB control key, which specifies a range of years around the year of birth to include in a search.

Syntax

```
public int getDobSearchRange()
```

Parameters

Parameter	Description
None	

Return Value

The **getDobSearchRange** method returns one of the following values:

This value is returned …	if this occurs …
Integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the SRCHDOB control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getDobSearchRange** to retrieve the value for the SRCHDOB control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
int range = contrlKey.getDobSearchRange()
System.out.println("DOB range: " +  );
```

getDuplicateSearchLimit

Description

The **getDuplicateSearchLimit** method retrieves the value of the PDSRCHLMT control key, which limits the number of records that can be returned by a potential duplicate search.

Syntax

```
public int getDuplicateSearchLimit ()
```

Parameters

Parameter	Description
None	

Return Value

The **getDuplicateSearchLimit** method returns one of the following values:

This value is returned …	if this occurs …
Integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the PDSRCHLMT control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getDuplicateSearchLimit** to retrieve the value for the PDSRCHLMT control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
int dupLimit = contrlKey.getDuplicateSearchLimit();
System.out.println("Dup limit: " + dupLimit);
```

getDuplicateThreshold

Description

The **getDuplicateThreshold** method retrieves the value of the DUPTHRES control key, which specifies the lowest matching weight at which two records are considered potential duplicates of one another.

Syntax

```
public float getDuplicateThreshold()
```

Parameters

Parameter	Description
None.	

Return Value

The **getDuplicateThreshold** method returns one of the following values:

This value is returned …	if this occurs …
Floating integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the DUPTHRES control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getDuplicateThreshold** to retrieve the value for the DUPTHRES control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
float dupThresh = contrlKey.getDuplicateThreshold();
System.out.println("Dup thresh: " + dupThresh);
```

getEndTime

Description

The **getEndTime** method retrieves the value of the ENDTIME control key, which specifies a default value for the "End Time" criteria for audit trail and potential duplicate searches.

Syntax

```
public java.lang.String getEndTime()
```

Parameters

Parameter	Description
None	

Return Value

The **getEndTime** method returns one of the following values:

This value is returned …	if this occurs …
String containing a control key value	The control key was checked successfully and its value was returned.
Null	No value was associated with the ENDTIME control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getEndTime** to retrieve the value for the ENDTIME control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
String endTime = contrlKey.getEndTime()
System.out.println("end time: " + endTime);
```

getExactSearchLength

Description

The **getExactSearchLength** method retrieves the value of the LNEXCTSRCH control key, which specifies the maximum number of letters that can be entered into the last name field in order to perform an exact match search against the criteria.

Syntax

```
public int getExactSearchLength()
```

Parameters

Parameter	Description
None	

Return Value

The **getExactSearchLength** method returns one of the following values:

This value is returned …	if this occurs …
Integer between 1 and 5, inclusive	The control key was checked successfully and its value was returned.
Null	No value was associated with the LNEXCTSRCH control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getExactSearchLength** to retrieve the value for the LNEXCTSRCH control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
int length = contrlKey.getExactSearchLength();
System.out.println("Exact last name search: " + length);
```

getInstance

Description

The **getInstance** method retrieves the single instance of the ControlKey class for the given **EiServer** object. This method is directly instantiated in a singleton pattern so database results are stored for servicing subsequent code lookups. This ensures that every transaction performed during a session uses the same control key configuration information.

Syntax

```
public static ControlKey getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	An EiServer object.

Return Value

The **getInstance** method returns one of the following values:

This value is returned …	if this occurs …
An instance of the ControlKey class	The instance was created successfully for the given EiServer object.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It the calls **ControlKey.getInstance(eiServer)** and to create an instance of the ControlKey class for the specified EiServer object.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
int uid = contrlKey.getUidLength();
System.out.println("UID Length: " + uid);
```

getMatchThreshold

Description

The **getMatchThreshold** method retrieves the value of the MATCHTHRES control key, which specifies the minimum matching weight at which two records are automatically merged.

Syntax

```
public float getMatchThreshold()
```

Parameters

Parameter	Description
None	

Return Value

The **getMatchThreshold** method returns one of the following values:

This value is returned …	if this occurs …
Floating integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the MATCHTHRES control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getMatchThreshold** to retrieve the value for the MATCHTHRES control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
float match = contrlKey.getMatchThreshold();
System.out.println("Match thresh: " + match);
```

getMaximumThreshold

Description

The **getMaximumThreshold** method retrieves the value of the MAXPROB control key, which specifies the largest matching weight value to be used.

Syntax

```
public float getMaximumThreshold()
```

Parameters

Parameter	Description
None	

Return Value

The **getMaximumThreshold** method returns one of the following values:

This value is returned …	if this occurs …
Floating integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the MAXPROB control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getMaximumThreshold** to retrieve the value for the MAXPROB control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
float max = contrlKey.getMaximumThreshold();
System.out.println("Max thresh: " + max);
```

getMinimumThreshold

Description

The **getMinimumThreshold** method retrieves the value of the THRESHOLD control key, which specifies the minimum matching weight for records returned from a search.

Syntax

```
public float getMinimumThreshold()
```

Parameters

Parameter	Description
None	

Return Value

The **getMinimumThreshold** method returns one of the following values:

This value is returned …	if this occurs …
Floating integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the THRESHOLD control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getMinimumThreshold** to retrieve the value for the THRESHOLD control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
float min = contrlKey.getMinimumThreshold();
System.out.println("Min thresh: " + min);
```

getStartTime

Description

The **getStartTime** method retrieves the value of the STARTTIME control key, which specifies the default value for the "Start Time" criteria for an audit trail or potential duplicate search.

Syntax

```
public java.lang.String getStartTime()
```

Parameters

Parameter	Description
None	

Return Value

The **getStartTime** method returns one of the following values:

This value is returned …	if this occurs …
String containing a control key value	The control key was checked successfully and its value was returned.
Null	No value was associated with the STARTTIME control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getStartTime** to retrieve the value for the STARTTIME control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
String startTime = contrlKey.getStartTime();
System.out.println("start time: " + startTime);
```

getUidLength

Description

The **getUidLength** method retrieves the value of the UIDLENGTH control key, which specifies the length of the unique identifier assigned by e*Index.

Syntax

```
public int getUidLength()
```

Parameters

Parameter	Description
None	

Return Value

The **getUidLength** method returns one of the following values:

This value is returned …	if this occurs …
Integer	The control key was checked successfully and its value was returned.
Null	No value was associated with the UIDLENGTH control key (this should not occur and indicates an error in the *ui_control* table).

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey. It then calls **getUidLength** to retrieve the value for the UIDLENGTH control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
int uid = contrlKey.getUidLength();
System.out.println("UID Length: " + uid);
```

isAllowNumericEnabled

Description

The **isAllowNumericEnabled** method retrieves the value of the ALLOWNUM control key, which specifies whether numeric characters are allowed in the **Last Name** and **First Name** fields.

Syntax

```
public boolean isAllowNumericEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isAllowNumericEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isAllowNumericEnabled** to retrieve the Boolean value for the ALLOWNUM control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isAllowNumericEnabled();
if (rc)
    System.out.println("Allow numeric: YES");
else
    System.out.println("Allow numeric: NO");
```

isAssumeMatchEnabled

Description

The **isAssumeMatchEnabled** method retrieves the value of the ASSMTCH control key, which specifies whether e*Index performs an automatic merge when the matching weight between two records is equal to or greater than the MATCHTHRES value.

Syntax

```
public boolean isAssumeMatchEnabled(java.lang.String key)
```

Parameters

Parameter	Description
None	

Return Value

The **isAssumeMatchEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isAssumeMatchEnabled** to retrieve the Boolean value for the ASSMTCH control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isAssumeMatchEnabled();
if (rc)
    System.out.println("Assume match: YES");
else
    System.out.println("Assume match: NO");
```

isAuditOnOff

Description

The **isAuditOnOff** method retrieves the value of the AUDITONOFF control key, which specifies whether to display the length of time required to perform each transaction.

Syntax

```
public boolean isAuditOnOff()
```

Parameters

Parameter	Description
None	

Return Value

The **isAuditOnOff** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isAuditOnOff** to retrieve the Boolean value for the AUDITONOFF control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isAuditOnOff();
if (rc)
    System.out.println("Audit On: YES");
else
    System.out.println("Audit On: NO");
```

isBlankOnUpdateEnabled

Description

The **isBlankOnUpdateEnabled** method retrieves the value of the BLNKONUPDT control key, which specifies whether a blank field should overwrite an existing value during an update transaction.

Syntax

```
public boolean isBlankOnUpdateEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isBlankOnUpdateEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isBlankOnUpdateEnabled** to retrieve the Boolean value for the BLNKONUPDT control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isBlankOnUpdateEnabled();
if (rc)
    System.out.println("Update On: YES");
else
    System.out.println("Update On: NO");
```

isDobYearRequired

Description

The **isDobYearRequired** method retrieves the value of the DOBYYREQ control key, which specifies whether a year of birth is required in order to perform a demographic search.

Syntax

```
public boolean isDobYearRequired()
```

Parameters

Parameter	Description
None	

Return Value

The **isDobYearRequired** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isDobYearRequired** to retrieve the Boolean value for the DOBYYREQ control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isDobYearRequired();
if (rc)
    System.out.println("DOB year required: YES");
else
    System.out.println("DOB year required: NO");
```

isDuplicateCheckEnabled

Description

The **isDuplicateCheckEnabled** method retrieves the value of the DUPCHK control key, which specifies whether a check for potential duplicates is performed each time a key field in a record is updated.

Syntax

```
public boolean isDuplicateCheckEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isDuplicateCheckEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It calls **isDuplicateCheckEnabled** to retrieve the Boolean value for the DUPCHK control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isDuplicateCheckEnabled();
if (rc)
    System.out.println("Duplicate check: YES");
else
    System.out.println("Duplicate Check: NO");
```

isExtensiveSearchEnabled

Description

The **isExtensiveSearchEnabled** method retrieves the value of the EXTNSVSRCH control key, which specifies whether searches include alias names.

Syntax

```
public boolean isExtensiveSearchEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isExtensiveSearchEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isExtensiveSearchEnabled** to retrieve the Boolean value for the EXTNSVSRCH control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isExtensiveSearchEnabled();
if (rc)
    System.out.println("Extensive search: YES");
else
    System.out.println("Extensive search: NO");
```

isMixedCaseEnabled

Description

The **isMixedCaseEnabled** method retrieves the value of the MIXEDCASE control key, which specifies whether data can be entered in mixed case into the e*Index GUIs.

Syntax

```
public boolean isMixedCaseEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isMixedCaseEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isMixedCaseEnabled** to retrieve the Boolean value for the MIXEDCASE control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isMixedCaseEnabled();
if (rc)
    System.out.println("Mixed case: YES");
else
    System.out.println("Mixed case: NO");
```

isOneExactMatchEnabled

Description

The **isOneExactMatchEnabled** method retrieves the value of the 1XACTMTCH control key, which specifies whether the record with the highest matching probability weight over the match threshold will be automatically merged with the new record (when there are multiple records over the match threshold).

Syntax

```
public boolean isOneExactMatchEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isOneExactMatchEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It calls **isOneExactMatchEnabled** to retrieve the Boolean value for the 1XACTMTCH control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isOneExactMatchEnabled();
if (rc)
    System.out.println("One Exact match: YES");
else
    System.out.println("One Exact match: NO");
```

isSameFacilityReportEnabled

Description

The **isSameFacilityReportEnabled** method retrieves the value of the SMEFACREP control key, which specifies whether a record is kept of person records that are flagged as duplicates from the same system.

Syntax

```
public boolean isSameFacilityReportEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isSameFacilityReportEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It calls **isSameFacilityReportEnabled** to retrieve the Boolean value for the SMEFACREP control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isSameFacilityReportEnabled();
if (rc)
    System.out.println("One Exact match: YES");
else
    System.out.println("One Exact match: NO");
```

isSeeDeactivatedEnabled

Description

The **isSeeDeactivatedEnabled** method retrieves the value of the SEEDEACTIV control key, which specifies whether deactivated person records are returned for a search.

Syntax

```
public boolean isSeeDeactivatedEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isSeeDeactivatedEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It calls **isSeeDeactivatedEnabled** to retrieve the Boolean value for the SEEDEACTIV control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isSeeDeactivatedEnabled();
if (rc)
    System.out.println("See Deactivated: YES");
else
    System.out.println("See Deactivated: NO");
```

isSeeMergedEnabled

Description

The **isSeeMergedEnabled** method retrieves the value of the SEEMERGED control key, which specifies whether records with a status of "Merged" are returned from a search.

Syntax

```
public boolean isSeeMergedEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isSeeMergedEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isSeeMergedEnabled** to retrieve the Boolean value for the SEEMERGED control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isSeeMergedEnabled();
if (rc)
    System.out.println("See Merged: YES");
else
    System.out.println("See Merged: NO");
```

isShortIdEnabled

Description

The **isShortIdEnabled** method retrieves the value of the SHORTID control key, which specifies whether the SSN field can be shorter than the value specified in the country-specific options in e*Index Administrator.

Syntax

```
public boolean isShortIdEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isShortIdEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The control key was checked successfully and its value is **Y**.
False	The control key was checked successfully and its value is **N**.

Throws

None.

Example

The following sample gets a new instance of EiServer and ControlKey, and then initializes the Boolean variable **rc**. It then calls **isShortIdEnabled** to retrieve the Boolean value for the SHORTID control key.

```
EiServer eiServer = new EiServer("EiServer.properties");
ControlKey contrlKey = ControlKey.getInstance(eiServer);
boolean rc = false;
rc = contrlKey.isShortIdEnabled();
if (rc)
    System.out.println("Short ID: YES");
else
    System.out.println("Short ID: NO");
```

CountryOption Class

Description

The **CountryOption** class represents the country-specific information stored in the *ui_misc_option* table. This information is defined in the Country Specific Option function of e*Index Administrator.

Properties

The **CountryOption** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.core.DataObject**

```
java.lang.Object
└─com.stc.eIndex.active.core.DataObject
    └─com.stc.eIndex.active.person.CountryOption
```

Constructor

None.

Methods

The methods included in the **CountryOption** class are described in detail on the following pages:

- **getControlType** on page 4-136
- **getCountryCode** on page 4-137
- **getOptionName** on page 4-138
- **getValue** on page 4-139

Inherited Methods

The **CountryOption** class inherits these methods from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **CountryOption** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

getControlType

Description

The **getControlType** method retrieves the value of the **controlType** field in a country-specific attribute. You can use this method to display control type code for a specific record in *ui_misc_option*.

Syntax

```
public java.lang.String getControlType()
```

Parameters

Parameter	Description
None	

Return Value

The **getControlType** method returns the following value:

This value is returned …	if this occurs …
String containing a control type code	The **controlType** field of the country-specific attribute was retrieved successfully.

Throws

None.

Example

The following example obtains an instance of EiServer and CountryOptionLookup. It then checks the input (*option_type*) to retrieve the enumeration for the corresponding field. The example calls **getOptionSet** in the CountryOptionLookup class to retrieve a vector of definitions for the specified control type. It then calls the '**get**' methods in the CountryOption class to display information about each record returned in the vector.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 CountryOptionLookup col = CountryOptionLookup.getInstance(eiServer);
 EnumCountryOption controlType = null;

 String field = option_type;
 if (field.equals("address"))
    controlType = EnumCountryOption.ADDRESS_SEARCH;
 else if (field.equals("column"))
    controlType = EnumCountryOption.COLUMN_FORMAT;
 else if (field.equals("tabs"))
    controlType = EnumCountryOption.TAB_LABEL;
 else if (field.equals("groups"))
    controlType = EnumCountryOption.GROUP_LABEL;
 else if (field.equals("summary"))
    controlType = EnumCountryOption.SUMMARY_TAB;

 Vector vector = col.getOptionSet(controlType);
 Enumeration optionSet = vector.elements();
 while (optionSet.hasMoreElements()) {
    CountryOption option = (CountryOption)optionSet.nextElement();
    System.out.println("Control Type:  "  + option.getControlType());
    System.out.println("Country Code:  "  + option.getCountryCode());
    System.out.println("Option Name:   "  + option.getOptionName());
    System.out.println("Option Value:  "  + option.getValue());
 }
...
```

getCountryCode

Description

The **getCountryCode** method retrieves the value of the **countryCode** field for country-specific attributes. You can use this method to display the county code for a specific record in *ui_misc_option*.

Syntax

```
public java.lang.String getCountryCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getCountryCode** method returns the following value:

This value is returned …	if this occurs …
String containing a country code	The **countryCode** field of the country-specific attribute was retrieved successfully.

Throws

None.

Example

To see an example of how **getCountryCode** can be used, see the example for **getControlType** beginning on page 4-136.

getOptionName

Description

The **getOptionName** method retrieves the value of the **optionName** field in a country-specific attribute. You can use this method to display the name of a country-specific attribute.

Syntax

```
public java.lang.String getOptionName()
```

Parameters

Parameter	Description
None	

Return Value

The **getOptionName** method returns the following value:

This value is returned …	if this occurs …
String containing an option name	The **optionName** field of the country-specific attribute was retrieved successfully.

Throws

None.

Example

To see an example of how **getOptionName** can be used, see the example for **getControlType** beginning on page 4-136.

getValue

Description

The **getValue** method retrieves the value of the **value** field in a country-specific attribute. You can use this method to display the value defined for the attribute.

Syntax

```
public java.lang.String getValue()
```

Parameters

Parameter	Description
None	

Return Value

The **getValue** method returns the following value:

This value is returned …	if this occurs …
String containing the value of the specified attribute	The **value** field of the country-specific attribute was retrieved successfully.

Throws

None.

Example

To see an example of how **getValue** can be used, see the example for **getControlType** beginning on page 4-136.

CountryOptionLookup

Description

The **CountryOptionLookup** class looks up country-specific attribute information by control type or it looks up the value of a country-specific attribute by control type, country, and option name.

Properties

The **CountryOptionLookup** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└com.stc.eIndex.active.person.CountryOptionLookup

Constructor

None.

Methods

The methods included in the **CountryOptionLookup** class are described in detail on the following pages:

- **getInstance** on page 4-141
- **getOptionSet** on page 4-141
- **lookupOptionValue** on page 4-142

Inherited Methods

The **CountryOptionLookup** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getInstance

Description

The **getInstance** method retrieves a static instance of the CountryOptionLookup class for the given EiServer object.

Syntax

```
public static CountryOptionLookup getInstance(EiServer
eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the CountryOptionLookup instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of CountryOptionLookup	An instance of the CountryOptionLookup class was retrieved successfully for the specified EiServer object.

Throws

The **getInstance** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQL**

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

To see an example of how **getInstance** can be used, see the example for **getControlType** beginning on page 4-136.

getOptionSet

Description

The **getOptionSet** method retrieves country-specific information for the specified control type. Control types are the categories of country-specific attributes, such as tab labels, address parsing, group labels, and so on.

Syntax

```
public java.util.Vector getOptionSet(EnumCountryOption
controlType)
```

Parameters

Parameter	Description
controlType	The type of country-specific option information you want to look up. This parameter should be the coded value of the control type. Possible values are defined in the **EnumCountryOption** class.

Return Value

The **getOptionSet** method returns the following value:

This value is returned …	if this occurs …
Vector of CountryOption objects	The country-specific information for the given control type was retrieved successfully.

Throws

The **getOptionSet** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQL**

Example

To see an example of how **getOptionSet** can be used, see the example for **getControlType** beginning on page 4-136.

lookupOptionValue

Description

The **lookupOptionValue** method retrieves the value of a country-specific attribute given the control type and option name. The value is retrieved for the country defined by the COUNTRY control key. The value is retrieved from the hash table.

Syntax

```
public java.lang.String lookupOptionValue(EnumCountryOption
controlType, java.lang.String optionName)
```

Parameters

Parameter	Description
controlType	The type of country-specific option information you want to look up. This parameter should be the coded value of the control type. Possible values are defined in the **EnumCountryOption** class.
optionName	The name of the country-specific attribute you want to look up. Possible values are defined in the option_name column of *ui_misc_option*.

Return Value

The **lookupOptionValue** method returns the following value:

This value is returned ...	if this occurs ...
String containing the option value	The value of the country-specific attribute was retrieved successfully.

Throws

The **lookupOptionValue** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQL**

Example

The following example obtains an instance of EiServer and CountryOptionLookup. It then retrieves an enumeration for the TAB_LABEL control type and calls **lookupOptionValue** to retrieve the value associated with the specified option name (*option_name*).

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 CountryOptionLookup lookup = CountryOptionLookup.getInstance(eiServer);
 EnumCountryOption conType = EnumCountryOption.TAB_LABEL;
 String value = lookup.lookupOptionValue(conType, option_name);
...
```

Demographics Class

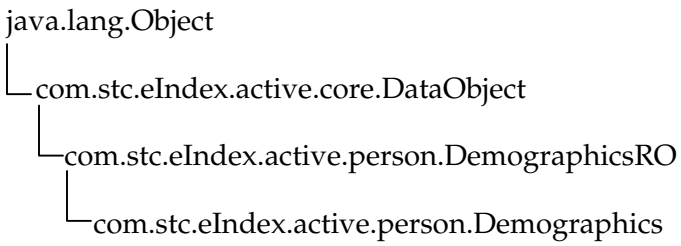
Description

The **Demographics** class represents the demographic information of a person record. The field values in this class can be modified after initialization.

Properties

The **Demographics** class has the following properties:

- Public class
- Direct known subclasses: **com.stc.eIndex.active.person.Person** and **com.stc.eIndex.active.person.SearchParameters**
- Extends **com.stc.eIndex.active.person.DemographicsRO**



Constructor

The **Demographics** class has one constructor, which is described on the following page:

- **Demographics** on page 4-147

Methods

The methods included in the **Demographics** class are described in detail on the following pages:

- **clearAll** on page 4-148
- **copyDemographics** on page 4-149
- **setCitizenship** on page 4-150
- **setClass1** on page 4-150
- **setClass2** on page 4-151
- **setClass3** on page 4-151
- **setClass4** on page 4-152
- **setClass5** on page 4-153
- **setDate1** on page 4-153
- **setDate2** on page 4-154
- **setDate3** on page 4-155
- **setDate4** on page 4-155

- setString5 on page 4-182
- setString6 on page 4-183
- setString7 on page 4-183
- setString8 on page 4-184
- setString9 on page 4-185
- setSuffix on page 4-185
- setTitle on page 4-185
- setVeteranStatus on page 4-186
- setVipFlag on page 4-187

Inherited Methods

The **Demographics** class inherits all methods defined in the class DemographicsRO Class (see page 4-242 for information about these Java methods).

The **Demographics** class also inherits these methods from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **Demographics** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Demographics

Description

The **Demographics** constructor method creates a new Demographic object containing the demographic information for a person record. This method can take either zero or one parameters. If given zero parameters, the constructor creates a new Demographic object, which can then be populated by the methods in this class. If given one parameter, a Demographic object, the constructor can be used to copy demographic information.

Syntax

```
public Demographics()
```

or

```
public Demographics(Demographics demo)
```

Parameters

Parameter	Description
demo	A list of demographic information. This parameter is optional.

Return Value

The **Demographics** constructor method returns the following value:

This value is returned …	if this occurs …
Demographic object	The demographic information was retrieved and the new Demographic object was created.

Throws

None.

Example

The following example creates an empty Demographic object **demo**, and then fills the object with demographic information using the '**set**' methods in the Demographics class. Once the Demographic object is created, the example copies the information into a second Demographics object, **demo2**.

Finally, all the properties in the original Demographics object, **demo**, are cleared by a call to **clearAll** so the object can be re-used.

```
...
 Demographics demo = new Demographics();
     demo.setLastName(last_name);
     demo.setFirstName(first_name);
     demo.setMiddleName(middle_name);
     demo.setSpouseName(spouse_name);
     demo.setMotherName(mother_name);
     demo.setFatherName(father_name);
     demo.setMaidenName(maiden_name);
     demo.setMotherMaidenName(mother_maiden_name);
     demo.setGender(gender_code);
     demo.setDob(date_of_birth);
     demo...

 Demographics demo2 = new Demographics(demo);
 demo.clearAll();
...
```

clearAll

Description

The **clearAll** method resets all demographic fields to uninitialized status. If you re-use a Demographic object, use this method to clear the demographic fields before each use.

Syntax

```
public void clearAll()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

None.

Example

For an example of how **clearAll** can be used, see the example for **Demographics** on page 4-147.

copyDemographics

Description

The **copyDemographics** method copies the demographic elements of another Demographics object to a new Demographics object. Since the **Demographics** and **Person** classes inherit from the **DemographicsRO** class, any of the three classes can be passed as a parameter to this function.

Syntax

```
public void copyDemographics(DemographicsRO demo)
```

or

```
public void copyDemographics(Demographics demo)
```

or

```
public void copyDemographics(Person demo)
```

Parameters

Parameter	Description
demo	A list of demographic information as represented by the Demographics, DemographicsRO, or Person class.

Return Value

None.

Throws

None.

Example

The following example creates a Demographics object **demo**, and then populates its properties using the '**set**' methods. It then creates a new Demographics object, **demo2**, and calls **copyDemographics** to copy the information from **demo** to **demo2**.

```
Demographics demo = new Demographics();
    demo.setLastName(last_name);
    demo.../* Populating property values */

    Demographics demo2 = new Demographics();
    demo2.copyDemographics(demo);
...
```

setCitizenship

Description

The **setCitizenship** method is the setter for the **citizenship** field. Use this method to populate the citizenship field in a Demographics object.

Syntax

```
public void setCitizenship(java.lang.String citizenship)
```

Parameters

Parameter	Description
citizenship	The value of the **citizenship** field for the new Demographics object.

Return Value

None.

Throws

The **setCitizenship** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setClass1

Description

The **setClass1** method is the setter for the **class1** field. Use this method to populate the class1 field in a Demographics object.

Syntax

```
public void setClass1(java.lang.String class1)
```

Parameters

Parameter	Description
class1	The value of the **class1** field for the new Demographics object.

Return Value

None.

Throws

The **setClass1** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setClass2

Description

The **setClass2** method is the setter for the **class2** field. Use this method to populate the class2 field in a Demographics object.

Syntax

```
public void setClass2(java.lang.String class2)
```

Parameters

Parameter	Description
class2	The value of the **class2** field for the new Demographics object.

Return Value

None.

Throws

The **setClass2** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setClass3

Description

The **setClass3** method is the setter for the **class3** field. Use this method to populate the class3 field in a Demographics object.

Syntax

```
public void setClass3(java.lang.String class3)
```

Parameters

Parameter	Description
class3	The value of the **class3** field for the new Demographics object.

Return Value

None.

Throws

The **setClass3** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setClass4

Description

The **setClass4** method is the setter for the **class4** field. Use this method to populate the class4 field in a Demographics object.

Syntax

```
public void setClass4(java.lang.String class4)
```

Parameters

Parameter	Description
class4	The value of the **class4** field for the new Demographics object.

Return Value

None.

Throws

The **setClass4** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setClass5

Description

The **setClass5** method is the setter for the **class5** field. Use this method to populate the class5 field in a Demographics object.

Syntax

```
public void setClass5(java.lang.String class5)
```

Parameters

Parameter	Description
class5	The value of the **class5** field for the new Demographics object.

Return Value

None.

Throws

The **setClass5** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDate1

Description

The **setDate1** method is the setter for the **date1** field. Use this method to populate the date1 field in a Demographics object.

Syntax

```
public void setDate1(java.sql.Date date1)
```

Parameters

Parameter	Description
date1	The value of the **date1** field for the new Demographics object.

Return Value

None.

Throws

The **setDate1** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDate2

Description

The **setDate2** method is the setter for the **date2** field. Use this method to populate the date2 field in a Demographics object.

Syntax

```
public void setDate2(java.sql.Date date2)
```

Parameters

Parameter	Description
date2	The value of the **date2** field for the new Demographics object.

Return Value

None.

Throws

The **setDate2** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDate3

Description

The **setDate3** method is the setter for the **date3** field. Use this method to populate the date3 field in a Demographics object.

Syntax

```
public void setDate3(java.sql.Date date3)
```

Parameters

Parameter	Description
date3	The value of the **date3** field for the new Demographics object.

Return Value

None.

Throws

The **setDate3** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDate4

Description

The **setDate4** method is the setter for the **date4** field. Use this method to populate the date4 field in a Demographics object.

Syntax

```
public void setDate4(java.sql.Date date4)
```

Parameters

Parameter	Description
date4	The value of the **date4** field for the new Demographics object.

Return Value

None.

Throws

The **setDate4** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDate5

Description

The **setDate5** method is the setter for the **date5** field. Use this method to populate the date5 field in a Demographics object.

Syntax

```
public void setDate5(java.sql.Date date5)
```

Parameters

Parameter	Description
date5	The value of the **date5** field for the new Demographics object.

Return Value

None.

Throws

The **setDate5** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDateOfDeath

Description

The **setDateOfDeath** method is the setter for the **dateOfDeath** field. Use this method to populate the date of death field in a Demographics object.

Syntax

```
public void setDateOfDeath(java.sql.Date dateOfDeath)
```

Parameters

Parameter	Description
dateOfDeath	The value of the **dateOfDeath** field for the new Demographics object.

Return Value

None.

Throws

The **setDateOfDeath** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDeath

Description

The **setDeath** method is the setter for the **death** field. Use this method to populate the death indicator in a Demographics object.

Syntax

```
public void setDeath(java.lang.String death)
```

Parameters

Parameter	Description
death	The value of the **death** field for the new Demographics object.

Return Value

None.

Throws

The **setDeath** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDeathCertificate

Description

The **setDeathCertificate** method is the setter for the **deathCertificate** field. Use this method to populate the death certificate indicator in a Demographics object.

Syntax

```
public void setDeath(java.lang.String deathCertificate)
```

Parameters

Parameter	Description
deathCertificate	The value of the **deathCertificate** field for the new Demographics object.

Return Value

None.

Throws

The **setDeathCertificate** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDistrictOf Residence

Description

The **setDistrictOfResidence** method is the setter for the **districtOfResidence** field. Use this method to populate the district of residence (DOR) in a Demographics object.

Syntax

```
public void setDistrictOfResidence(java.lang.String
districtOfResidence)
```

Parameters

Parameter	Description
districtOfResidence	The value of the **districtOfResidence** field for the new Demographics object.

Return Value

None.

Throws

The **setDistrictOfResidence** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDob

Description

The **setDob** method is the setter for the **dob** field. Use this method to populate the date of birth in a Demographics object.

Syntax

```
public void setDob(java.sql.Date dob)
```

Parameters

Parameter	Description
dob	The value of the **dob** field for the new Demographics object.

Return Value

None.

Throws

The **setDob** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDriversLicenseNumber

Description

The **setDriversLicenseNumber** method is the setter for the **driversLicenseNumber** field. Use this method to populate the drivers license in a Demographics object.

Syntax

```
public void setDriversLicenseNumber(java.lang.String
driversLicenseNumber)
```

Parameters

Parameter	Description
driversLicenseNumber	The value of the **driversLicenseNumber** field for the new Demographics object.

Return Value

None.

Throws

The **setDriversLicenseNumber** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setDriversLicenseState

Description

The **setDriversLicenseState** method is the setter for the **driversLicenseState** field. Use this method to populate the state that issued the drivers license in a Demographics object.

Syntax

```
public void setDriversLicenseState(java.lang.String
driversLicenseState)
```

Parameters

Parameter	Description
driversLicenseState	The value of the **driversLicenseState** field for the new Demographics object.

Return Value

None.

Throws

The **setDriversLicenseState** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setEthnic

Description

The **setEthnic** method is the setter for the **ethnic** field. Use this method to populate the ethnic group in a Demographics object.

Syntax

```
public void setEthnic(java.lang.String ethnic)
```

Parameters

Parameter	Description
ethnic	The value of the **ethnic** field for the new Demographics object.

Return Value

None.

Throws

The **setEthnic** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setFatherName

Description

The **setFatherName** method is the setter for the **fatherName** field. Use this method to populate the father name in a Demographics object.

Syntax

```
public void setFatherName(java.lang.String fatherName)
```

Parameters

Parameter	Description
fatherName	The value of the **fatherName** field for the new Demographics object.

Return Value

None.

Throws

The **setFatherName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setFirstName

Description

The **setFirstName** method is the setter for the **firstName** field. Use this method to populate the first name in a Demographics object.

Syntax

```
public void setFirstName(java.lang.String firstName)
```

Parameters

Parameter	Description
firstName	The value of the **firstName** field for the new Demographics object.

Return Value

None.

Throws

The **setFirstName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setGender

Description

The **setGender** method is the setter for the **gender** field. Use this method to populate the gender in a Demographics object.

Syntax

```
public void setGender(java.lang.String gender)
```

Parameters

Parameter	Description
gender	The value of the **gender** field for the new Demographics object.

Return Value

None.

Throws

The **setGender** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setLanguage

Description

The **setLanguage** method is the setter for the **language** field. Use this method to populate the language in a Demographics object.

Syntax

```
public void setLanguage(java.lang.String language)
```

Parameters

Parameter	Description
language	The value of the **language** field for the new Demographics object.

Return Value

None.

Throws

The **setLanguage** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setLastName

Description

The **setLastName** method is the setter for the **lastName** field. Use this method to populate the last name in a Demographics object.

Syntax

```
public void setLastName(java.lang.String lastName)
```

Parameters

Parameter	Description
lastName	The value of the **lastName** field for the new Demographics object.

Return Value

None.

Throws

The **setLastName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setLgaCode

Description

The **setLgaCode** method is the setter for the **lgaCode** field. Use this method to populate the LGA code in a Demographics object.

Syntax

```
public void setLgaCode(java.lang.String lgaCode)
```

Parameters

Parameter	Description
lgaCode	The value of the **lgaCode** field for the new Demographics object.

Return Value

None.

Throws

The **setLgaCode** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMaidenName

Description

The **setMaidenName** method is the setter for the **maidenName** field. Use this method to populate the maiden name in a Demographics object.

Syntax

```
public void setMaidenName(java.lang.String maidenName)
```

Parameters

Parameter	Description
maidenName	The value of the **maidenName** field for the new Demographics object.

Return Value

None.

Throws

The **setMaidenName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMaritalStatus

Description

The **setMaritalStatus** method is the setter for the **maritalStatus** field. Use this method to populate the marital status in a Demographics object.

Syntax

```
public void setMaritalStatus(java.lang.String maritalStatus)
```

Parameters

Parameter	Description
maritalStatus	The value of the **maritalStatus** field for the new Demographics object.

Return Value

None.

Throws

The **setMaritalStatus** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMiddleName

Description

The **setMiddleName** method is the setter for the **middleName** field. Use this method to populate the middle name or initial of the name in a Demographics object.

Syntax

```
public void setMiddleName(java.lang.String middleName)
```

Parameters

Parameter	Description
middleName	The value of the **middleName** field for the new Demographics object.

Return Value

None.

Throws

The **setMiddleName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMilitaryBranch

Description

The **setMilitaryBranch** method is the setter for the **militaryBranch** field. Use this method to populate the military branch of the person represented in a Demographics object.

Syntax

```
public void setMilitaryBranch(java.lang.String militaryBranch)
```

Parameters

Parameter	Description
militaryBranch	The value of the **militaryBranch** field for the new Demographics object.

Return Value

None.

Throws

The **setMilitaryBranch** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMilitaryRank

Description

The **setMilitaryRank** method is the setter for the **militaryRank** field. Use this method to populate the military rank of the person represented in a Demographics object.

Syntax

```
public void setMilitaryRank(java.lang.String militaryRank)
```

Parameters

Parameter	Description
militaryRank	The value of the **militaryRank** field for the new Demographics object.

Return Value

None.

Throws

The **setMilitaryRank** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMilitaryStatus

Description

The **setMilitaryStatus** method is the setter for the **militaryStatus** field. Use this method to populate the military status of the person represented in a Demographics object.

Syntax

```
public void setMilitaryStatus(java.lang.String militaryStatus)
```

Parameters

Parameter	Description
militaryStatus	The value of the **militaryStatus** field for the new Demographics object.

Return Value

None.

Throws

The **setMilitaryStatus** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMotherMaidenName

Description

The **setMotherMaidenName** method is the setter for the **motherMaidenName** field. Use this method to populate the mother's maiden name in a Demographics object.

Syntax

```
public void setMotherMaidenName(java.lang.String
motherMaidenName)
```

Parameters

Parameter	Description
motherMaidenName	The value of the **motherMaidenName** field for the new Demographics object.

Return Value

None.

Throws

The **setMotherMaidenName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setMotherName

Description

The **setMotherName** method is the setter for the **motherName** field. Use this method to populate the mother's name in a Demographics object.

Syntax

```
public void setMotherName(java.lang.String motherName)
```

Parameters

Parameter	Description
motherName	The value of the **motherName** field for the new Demographics object.

Return Value

None.

Throws

The **setMotherName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setNationality

Description

The **setNationality** method is the setter for the **nationality** field. Use this method to populate the nationality in a Demographics object.

Syntax

```
public void setNationality(java.lang.String nationality)
```

Parameters

Parameter	Description
nationality	The value of the **nationality** field for the new Demographics object.

Return Value

None.

Throws

The **setNationality** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPensionExpirationDate

Description

The **setPensionExpirationDate** method is the setter for the **pensionExpirationDate** field. Use this method to populate the expiration date in a Demographics object.

Syntax

```
public void setPensionExpirationDate(java.lang.String
pensionExpirationDate)
```

Parameters

Parameter	Description
pensionExpirationDate	The value of the **pensionExpirationDate** field for the new Demographics object.

Return Value

None.

Throws

The **setPensionExpirationDate** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPensionNumber

Description

The **setPensionNumber** method is the setter for the **pensionNumber** field. Use this method to populate the pension number in a Demographics object.

Syntax

```
public void setPensionNumber(java.lang.String pensionNumber)
```

Parameters

Parameter	Description
pensionNumber	The value of the **pensionNumber** field for the new Demographics object.

Return Value

None.

Throws

The **setPensionNumber** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPersonCategoryCode

Description

The **setPersonCategoryCode** method is the setter for the **personCategoryCode** field. Use this method to populate the person category in a Demographics object.

Syntax

```
public void setPersonCategoryCode(java.lang.String
personCategoryCode)
```

Parameters

Parameter	Description
personCategoryCode	The value of the **personCategoryCode** field for the new Demographics object.

Return Value

None.

Throws

The **setPersonCategoryCode** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPobCity

Description

The **setPobCity** method is the setter for the **pobCity** field. Use this method to populate the city of birth in a Demographics object.

Syntax

```
public void setPobCity(java.lang.String pobCity)
```

Parameters

Parameter	Description
pobCity	The value of the **pobCity** field for the new Demographics object.

Return Value

None.

Throws

The **setPobCity** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPobCountry

Description

The **setPobCountry** method is the setter for the **pobCountry** field. Use this method to populate the country of birth in a Demographics object.

Syntax

```
public void setPobCountry(java.lang.String pobCountry)
```

Parameters

Parameter	Description
pobCountry	The value of the **pobCountry** field for the new Demographics object.

Return Value

None.

Throws

The **setCountry** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setPobState

Description

The **setPobState** method is the setter for the **pobState** field. Use this method to populate the state of birth in a Demographics object.

Syntax

```
public void setPobState(java.lang.String pobState)
```

Parameters

Parameter	Description
pobState	The value of the **pobState** field for the new Demographics object.

Return Value

None.

Throws

The **setPobState** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setRace

Description

The **setRace** method is the setter for the **race** field. Use this method to populate the race in a Demographics object.

Syntax

```
public void setRace(java.lang.String race)
```

Parameters

Parameter	Description
race	The value of the **race** field for the new Demographics object.

Return Value

None.

Throws

The **setRace** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setReligion

Description

The **setReligion** method is the setter for the **religion** field. Use this method to populate the religious denomination in a Demographics object.

Syntax

```
public void setReligion(java.lang.String religion)
```

Parameters

Parameter	Description
religion	The value of the **religion** field for the new Demographics object.

Return Value

None.

Throws

The **setReligion** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setRepatriationNumber

Description

The **setRepatriationNumber** method is the setter for the **repatriationNumber** field. Use this method to populate the repatriation number in a Demographics object.

Syntax

```
public void setRepatriationNumber(java.lang.String
repatriationNumber)
```

Parameters

Parameter	Description
repatriationNumber	The value of the **repatriationNumber** field for the new Demographics object.

Return Value

None.

Throws

The **setRepatriationNumber** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setSpouseName

Description

The **setSpouseName** method is the setter for the **spouseName** field. Use this method to populate the name of the spouse in a Demographics object.

Syntax

```
public void setSpouseName(java.lang.String spouseName)
```

Parameters

Parameter	Description
spouseName	The value of the **spouseName** field for the new Demographics object.

Return Value

None.

Throws

The **setSpouseName** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setSsn

Description

The **setSsn** method is the setter for the **ssn** field. Use this method to populate the social security number in a Demographics object.

Syntax

```
public void setSsn(Ssn ssn)
```

Parameters

Parameter	Description
ssn	The value of the **ssn** field for the new Demographics object.

Return Value

None.

Throws

The **setSsn** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Ssn Class**, see page 4-478.

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString1

Description

The **setString1** method is the setter for the **string1** field. Use this method to populate the value of the string1 field in a Demographics object.

Syntax

```
public void setString1(java.lang.String string1)
```

Parameters

Parameter	Description
string1	The value of the **string1** field for the new Demographics object.

Return Value

None.

Throws

The **setString1** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString10

Description

The **setString10** method is the setter for the **string10** field. Use this method to populate the value of the string10 field in a Demographics object.

Syntax

```
public void setString10(java.lang.String string10)
```

Parameters

Parameter	Description
string10	The value of the **string10** field for the new Demographics object.

Return Value

None.

Throws

The **setString10** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString2

Description

The **setString2** method is the setter for the **string2** field. Use this method to populate the value of the string2 field in a Demographics object.

Syntax

```
public void setString2(java.lang.String string2)
```

Parameters

Parameter	Description
string2	The value of the **string2** field for the new Demographics object.

Return Value

None.

Throws

The **setString2** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString3

Description

The **setString3** method is the setter for the **string3** field. Use this method to populate the value of the string3 field in a Demographics object.

Syntax

```
public void setString3(java.lang.String string3)
```

Parameters

Parameter	Description
string3	The value of the **string3** field for the new Demographics object.

Return Value

None.

Throws

The **setString3** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString4

Description

The **setString4** method is the setter for the **string4** field. Use this method to populate the value of the string4 field in a Demographics object.

Syntax

```
public void setString4(java.lang.String string4)
```

Parameters

Parameter	Description
string4	The value of the **string4** field for the new Demographics object.

Return Value

None.

Throws

The **setString4** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString5

Description

The **setString5** method is the setter for the **string5** field. Use this method to populate the value of the string5 field in a Demographics object.

Syntax

```
public void setString5(java.lang.String string5)
```

Parameters

Parameter	Description
string5	The value of the **string5** field for the new Demographics object.

Return Value

None.

Throws

The **setString5** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString6

Description

The **setString6** method is the setter for the **string6** field. Use this method to populate the value of the string6 field in a Demographics object.

Syntax

```
public void setString6(java.lang.String string6)
```

Parameters

Parameter	Description
string6	The value of the **string6** field for the new Demographics object.

Return Value

None.

Throws

The **setString6** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString7

Description

The **setString7** method is the setter for the **string7** field. Use this method to populate the value of the string7 field in a Demographics object.

Syntax

```
public void setString7(java.lang.String string7)
```

Parameters

Parameter	Description
string7	The value of the **string7** field for the new Demographics object.

Return Value

None.

Throws

The **setString7** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString8

Description

The **setString8** method is the setter for the **string8** field. Use this method to populate the value of the string8 field in a Demographics object.

Syntax

```
public void setString8(java.lang.String string8)
```

Parameters

Parameter	Description
string8	The value of the **string8** field for the new Demographics object.

Return Value

None.

Throws

The **setString8** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setString9

Description

The **setString9** method is the setter for the **string9** field. Use this method to populate the value of the string9 field in a Demographics object.

Syntax

```
public void setString9(java.lang.String string9)
```

Parameters

Parameter	Description
string9	The value of the **string9** field for the new Demographics object.

Return Value

None.

Throws

The **setString9** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setSuffix

Description

The **setSuffix** method is the setter for the **suffix** field. Use this method to populate the suffix to the name in a Demographics object.

Syntax

```
public void setSuffix(java.lang.String suffix)
```

Parameters

Parameter	Description
suffix	The value of the **suffix** field for the new Demographics object.

Return Value

None.

Throws

The **setSuffix** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setTitle

Description

The **setTitle** method is the setter for the **title** field. Use this method to populate the title to the name in a Demographics object.

Syntax

```
public void setTitle(java.lang.String title)
```

Parameters

Parameter	Description
title	The value of the **title** field for the new Demographics object.

Return Value

None.

Throws

The **setTitle** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setVeteranStatus

Description

The **setVeteranStatus** method is the setter for the **veteranStatus** field. Use this method to populate the veteran status in a Demographics object.

Syntax

```
public void setVeteranStatus(java.lang.String veteranStatus)
```

Parameters

Parameter	Description
veteranStatus	The value of the **veteranStatus** field for the new Demographics object.

Return Value

None.

Throws

The **setVeteranStatus** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

setVipFlag

Description

The **setVipFlag** method is the setter for the **vipFlag** field. Use this method to populate the VIP status of the person represented in a Demographics object.

Syntax

```
public void setVipFlag(java.lang.String vipFlag)
```

Parameters

Parameter	Description
vipFlag	The value of the **vipFlag** field for the new Demographics object.

Return Value

None.

Throws

The **setVipFlag** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Demographics object, see the example for **Demographics** on page 4-147.

DemographicsResultRO Class

Description

The **DemographicsResultRO** class represents a set of a person's demographic data. This data is returned as a result of an alphanumeric, phonetic, or general search, which all return a PersonSearchResult object representing a set of DemographicsResultRO objects. Once you retrieve a DemographicsResultRO object, you can use its UID to load the entire Person object, or you can call **getDemographics** (in PersonBO) to retrieve a DemographicsRO object. The Person object can be modified; the DemographicsRO object is read-only. Field values for the DemographicsResultRO class cannot be changed once initialized.

Important! *While the methods in the DemographicsResultRO class allow you to retrieve most demographic fields, the fields you can retrieve are limited by the configurable query (for more information, see chapter 5 in the* e*Index Administrator User's Guide*). You can only access a field in a DemographicsResultRO object if it is selected in the configurable query. Attempting to access fields not defined in the configurable query will result in a runtime exception, which aborts the thread and signals a fatal application error.*

Properties

The **DemographicsResultRO** class has the following properties:

- Public class
- Implements **java.lang.Comparable**
- Extends **com.stc.eIndex.active.core.DynamicDataObject**

```
java.lang.Object
└─com.stc.eIndex.active.core.DynamicDataObject
    └─com.stc.eIndex.active.person.DemographicsResultRO
```

Constructor

None.

Methods

The methods included in the **DemographicsResultRO** class are described in detail on the following pages:

Inherited Methods

The **DemographicsResultRO** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

compareTo

Description

The **compareTo** method compares two Demographics objects based on their probabilistic matching weights. This method is only used internally by the API to rank matching records returned from a search.

getCitizenship

Description

The **getCitizenship** method retrieves the value of a record's **citizenship** field. Use this method to display the citizenship for a person record returned from a demographic search.

Syntax

```
public java.lang.String getCitizenship()
```

Parameters

Parameter	Description
None	

Return Value

The **getCitizenship** method returns one of the following values:

This value is returned …	if this occurs …
String containing a citizenship code	The value of the **citizenship** field was retrieved successfully.
Null	The **citizenship** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getClass1

Description

The **getClass1** method retrieves the value of a record's **class1** field. Use this method to display the value of the **class1** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getClass1()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass1** method returns one of the following values:

This value is returned …	if this occurs …
String containing the value of **class1**	The value of the **class1** field was retrieved successfully.
Null	The **class1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getClass2

Description

The **getClass2** method retrieves the value of a record's **class2** field. Use this method to display the value of the **class2** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getClass2()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass2** method returns one of the following values:

This value is returned …	if this occurs …
String containing the value of **class2**	The value of the **class2** field was retrieved successfully.
Null	The **class2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getClass3

Description

The **getClass3** method retrieves the value of a record's **class3** field. Use this method to display the value of the **class3** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getClass3()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass3** method returns one of the following values:

This value is returned …	if this occurs …
String containing the value of **class3**	The value of the **class3** field was retrieved successfully.
Null	The **class3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getClass4

Description

The **getClass4** method retrieves the value of a record's **class4** field. Use this method to display the value of the **class4** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getClass4()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass4** method returns one of the following values:

This value is returned …	if this occurs …
String containing the value of **class4**	The value of the **class4** field was retrieved successfully.
Null	The **class4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getClass5

Description

The **getClass5** method retrieves the value of a record's **class5** field. Use this method to display the value of the **class5** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getClass5()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass5** method returns one of the following values:

This value is returned …	if this occurs …
String containing the value of **class5**	The value of the **class5** field was retrieved successfully.
Null	The **class5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getComparisonScore

Description

The **getComparisonScore** method retrieves the value of the **comparisonScore** property. This value represents the matching probability weight between the search criteria and each record returned by the search.

Syntax

```
public float getComparisonScore()
```

Parameters

Parameter	Description
None	

Return Value

The **getComparisonScore** method returns one of the following values:

This value is returned …	if this occurs …
Float representing the matching weight	The **comparisonScore** property was retrieved successfully.

Throws

None.

Example

To see an example of how **getComparisonScore** can be used, see the example for **getFirstName** beginning on page 4-208.

getDate1

Description

The **getDate1** method retrieves the value of a record's **date1** field. Use this method to display the value of the **date1** field in a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDate1()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate1** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date1** field was retrieved successfully.
Null	The **date1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDate2

Description

The **getDate2** method retrieves the value of a record's **date2** field. Use this method to display the value of the **date2** field in a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDate2()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate2** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date2** field was retrieved successfully.
Null	The **date2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDate3

Description

The **getDate3** method retrieves the value of a record's **date3** field. Use this method to display the value of the **date3** field in a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDate3()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate3** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date3** field was retrieved successfully.
Null	The **date3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDate4

Description

The **getDate4** method retrieves the value of a record's **date4** field. Use this method to display the value of the **date4** field in a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDate4()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate4** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date4** field was retrieved successfully.
Null	The **date4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDate5

Description

The **getDate5** method retrieves the value of a record's **date5** field. Use this method to display the value of the **date5** field in a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDate5()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate5** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date5** field was retrieved successfully.
Null	The **date5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDateOfDeath

Description

The **getDateOfDeath** method retrieves the value of a record's **dateOfDeath** field. Use this method to display the date of death for a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDateOfDeath()
```

Parameters

Parameter	Description
None	

Return Value

The **getDateOfDeath** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **dateOfDeath** field was retrieved successfully.

This value is returned …	if this occurs …
Null	The **dateOfDeath** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDeath

Description

The **getDeath** method retrieves the value of a record's **death** field. Use this method to display the value of the value of the **death** field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getDeath()
```

Parameters

Parameter	Description
None	

Return Value

The **getDeath** method returns one of the following values:

This value is returned …	if this occurs …
String containing a death indicator	The value of the **death** field was retrieved successfully.
Null	The **death** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDeathCertificate

Description

The **getDeathCertificate** method retrieves the value of a record's **deathCertificate** field. Use this method to display the death certificate number for a person record returned from a demographic search.

Syntax

```
public java.lang.String getDeathCertificate()
```

Parameters

Parameter	Description
None	

Return Value

The **getDeathCertificate** method returns one of the following values:

This value is returned …	if this occurs …
String containing death certificate information	The value of the **deathCertificate** field was retrieved successfully.
Null	The **deathCertificate** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDistrictOfResidence

Description

The **getDistrictOfResidence** method retrieves the value of a record's **districtOfResidence** field. Use this method to display the district of residence (DOR) for a person record returned from a demographic search.

Syntax

```
public java.lang.String getDistrictOfResidence()
```

Parameters

Parameter	Description
None	

Return Value

The **getDistrictOfResidence** method returns one of the following values:

This value is returned …	if this occurs …
String containing a DOR code	The value of the **districtOfResidence** field was retrieved successfully.
Null	The **districtOfResidence** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDob

Description

The **getDob** method retrieves the value of a record's **dob** field. Use this method to display the date of birth for a person record returned from a demographic search.

Syntax

```
public java.sql.Date getDob()
```

Parameters

Parameter	Description
None	

Return Value

The **getDob** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **dob** field was retrieved successfully.
Null	The **dob** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDriversLicenseNumber

Description

The **getDriversLicenseNumber** method retrieves the value of a record's **driversLicenseNumber** field. Use this method to display the driver license number for a person record returned from a demographic search.

Syntax

```
public java.lang.String getDriversLicenseNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getDriversLicenseNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a driver license number	The value of the **driversLicenseNumber** field was retrieved successfully.
Null	The **driversLicenseNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getDriversLicenseState

Description

The **getDriversLicenseState** method retrieves the value of a record's **driversLicenseState** field. Use this method to display state that issued the driver license for a person record returned from a demographic search.

Syntax

```
public java.lang.String getDriversLicenseState()
```

Parameters

Parameter	Description
None	

Return Value

The **getDriversLicenseState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **driversLicenseState** field was retrieved successfully.
Null	The **driversLicenseState** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getEthnic

Description

The **getEthnic** method retrieves the value of a record's **ethnic** field. Use this method to display the ethnicity for a person record returned from a demographic search.

Syntax

```
public java.lang.String getEthnic()
```

Parameters

Parameter	Description
None	

Return Value

The **getEthnic** method returns one of the following values:

This value is returned …	if this occurs …
String containing an ethnic code	The value of the **ethnic** field was retrieved successfully.
Null	The **ethnic** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getFatherName

Description

The **getFatherName** method retrieves the value of a record's **fatherName** field. Use this method to display the father's name for a person record returned from a demographic search.

Syntax

```
public java.lang.String getFatherName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFatherName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **fatherName** field was retrieved successfully.
Null	The **fatherName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getFirstName

Description

The **getFirstName** method retrieves the value of a record's **firstName** field. Use this method to display the first name in a person record returned from a demographic search.

Syntax

```
public java.lang.String getFirstName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFirstName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a first name	The value of the **firstName** field was retrieved successfully.
Null	The **firstName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

The example on the following page performs an alphanumeric search, returning the resulting records in a PersonSearchResult object named **psr**. Each record in **psr** is a DemographicsResultRO object, represented by the variable **dr**. Once the search is performed, the example scrolls through each record in **psr**, and calls the '**get**' methods in the DemographicsResultRO class to retrieve and print the individual properties for each **dr** object.

```
.../* Instantiating EiServer and creating
      SearchParameters object searchParams */

 PersonSearchResult psr = null;
 DemographicsResultRO dr = null;
 psr = personBO.searchAlpha(searchParams);
 SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

 if (psr==null) {
   System.out.println("No matching records.");
 } else {
   while(psr.hasNext()) {
     dr = psr.next();
      String dob = sdf.format(dr.getDob());
      CodeLookup cl = CodeLookup.getInstance(eiServer);
      String display = cl.getDisplayValue(EnumCodeType.GENDER,dr.getGender());
      System.out.println("UID:                " + dr.getUid());
      System.out.println("First Name:         " + dr.getFirstName());
      System.out.println("Middle Name:        " + dr.getMiddleName());
      System.out.println("Last Name:          " + dr.getLastName());
      System.out.println("Gender:             " + display);
      System.out.println("SSN:                " + dr.getSsn());
      System.out.println("Date of Birth:      " + dob);
      System.out.println("Street Address 1:   " + dr.getAddressStreet1());
      System.out.println("Status:             " + dr.getStatus());
      System.out.println ...
          /* Displaying remaining property values using the 'get' methods */
      System.out.println("Matching Weight:    " + dr.getComparisonScore());
...
```

getGender

Description

The **getGender** method retrieves the value of a record's **gender** field. Use this method to display the gender field in a person record returned from a demographic search.

Syntax

```
public java.lang.String getGender()
```

Parameters

Parameter	Description
None	

Return Value

The **getGender** method returns one of the following values:

This value is returned …	if this occurs …
String containing a gender	The value of the **gender** field was retrieved successfully.
Null	The **gender** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getLastName

Description

The **getLastName** method retrieves the value of a record's **lastName** field. Use this method to display the last name in a person record returned from a demographic search.

Syntax

```
public java.lang.String getLastName()
```

Parameters

Parameter	Description
None	

Return Value

The **getLastName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a last name	The value of the **lastName** field was retrieved successfully.
Null	The **lastName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getLgaCode

Description

The **getLgaCode** method retrieves the value of a record's **lgaCode** field. Use this method to display the LGA code for a person record returned from a demographic search.

Syntax

```
public java.lang.String getLgaCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getLgaCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing an LGA code	The value of the **lgaCode** field was retrieved successfully.
Null	The **lgaCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMaidenName

Description

The **getMaidenName** method retrieves the value of a record's **maidenName** field. Use this method to display the maiden name for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMaidenName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMaidenName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a last name	The value of the **maidenName** field was retrieved successfully.
Null	The **maidenName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMaritalStatus

Description

The **getMaritalStatus** method retrieves the value of a record's **maritalStatus** field. Use this method to display the marital status for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMaritalStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getMaritalStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing a status code	The value of the **maritalStatus** field was retrieved successfully.
Null	The **maritalStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMiddleName

Description

The **getMiddleName** method retrieves the value of a record's **middleName** field. Use this method to display the middle name or initial in a person record returned from a demographic search.

Syntax

```
public java.lang.String getMiddleName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMiddleName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a middle name	The value of the **middleName** field was retrieved successfully.
Null	The **middleName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMilitaryBranch

Description

The **getMilitaryBranch** method retrieves the value of a record's **militaryBranch** field. Use this method to display the military branch for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMilitaryBranch()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryBranch** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military branch	The value of the **militaryBranch** field was retrieved successfully.
Null	The **militaryBranch** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMilitaryRank

Description

The **getMilitaryRank** method retrieves the value of a record's **militaryRank** field. Use this method to display the military rank for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMilitaryRank()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryRank** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military rank	The value of the **militaryRank** field was retrieved successfully.
Null	The **militaryRank** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMilitaryStatus

Description

The **getMilitaryStatus** method retrieves the value of a record's **militaryStatus** field. Use this method to display the military status for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMilitaryStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military status	The value of the **militaryStatus** field was retrieved successfully.
Null	The **militaryStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getMotherMaidenName

Description

The **getMotherMaidenName** method retrieves the value of a record's **motherMaidenName** field. Use this method to display the mother's maiden name for a person record returned from a demographic search.

Syntax

```
public java.lang.String getMotherMaidenName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMotherMaidenName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a last name	The value of the **motherMaidenName** field was retrieved successfully.
Null	The **motherMaidenName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getNationality

Description

The **getNationality** method retrieves the value of a record's **nationality** field. Use this method to display the nationality for a person record returned from a demographic search.

Syntax

```
public java.lang.String getNationality()
```

Parameters

Parameter	Description
None	

Return Value

The **getNationality** method returns one of the following values:

This value is returned …	if this occurs …
String containing a nationality code	The value of the **nationality** field was retrieved successfully.
Null	The **nationality** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPensionExpirationDate

Description

The **getPensionExpirationDate** method retrieves the value of a record's **pensionExpirationDate** field. Use this method to display the expiration date of a pension for a person record returned from a demographic search.

Syntax

```
public java.sql.Date getPensionExpirationDate()
```

Parameters

Parameter	Description
None	

Return Value

The **getPensionExpirationDate** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **pensionExpirationDate** field was retrieved successfully.
Null	The **pensionExpirationDate** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPensionNumber

Description

The **getPensionNumber** method retrieves the value of a record's **pensionNumber** field. Use this method to display the pension number for a person record returned from a demographic search.

Syntax

```
public java.lang.String getPensionNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getPensionNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a pension number	The value of the **pensionNumber** field was retrieved successfully.
Null	The **pensionNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPersonCategoryCode

Description

The **getPersonCategoryCode** method retrieves the value of a record's **personCategoryCode** field. Use this method to display the person category code assigned to a person record returned from a demographic search.

Syntax

```
public java.lang.String getPersonCategoryCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getPersonCategoryCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a person category code	The value of the **personCategoryCode** field was retrieved successfully.
Null	The **personCategoryCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPobCity

Description

The **getPobCity** method retrieves the value of a record's **pobCity** field. Use this method to display the city of birth for a person record returned from a demographic search.

Syntax

```
public java.lang.String getPobCity()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobCity** method returns one of the following values:

This value is returned …	if this occurs …
String containing a city name	The value of the **pobCity** field was retrieved successfully.
Null	The **pobCity** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPobCountry

Description

The **getPobCountry** method retrieves the value of a record's **pobCountry** field. Use this method to display the country of birth for a person record returned from a demographic search.

Syntax

```
public java.lang.String getPobCountry()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobCountry** method returns one of the following values:

This value is returned …	if this occurs …
String containing a country code	The value of the **pobCountry** field was retrieved successfully.
Null	The **pobCountry** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getPobState

Description

The **getPobState** method retrieves the value of a record's **pobState** field. Use this method to display the state of birth for a person record returned from a demographic search.

Syntax

```
public java.lang.String getPobState()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **pobState** field was retrieved successfully.
Null	The **pobState** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getRace

Description

The **getRace** method retrieves the value of a record's **race** field. Use this method to display the race for a person record returned from a demographic search.

Syntax

```
public java.lang.String getRace()
```

Parameters

Parameter	Description
None	

Return Value

The **getRace** method returns one of the following values:

This value is returned …	if this occurs …
String containing a race code	The value of the **race** field was retrieved successfully.
Null	The **race** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getReligion

Description

The **getReligion** method retrieves the value of a record's **religion** field. Use this method to display the religion for a person record returned from a demographic search.

Syntax

```
public java.lang.String getReligion()
```

Parameters

Parameter	Description
None	

Return Value

The **getReligion** method returns one of the following values:

This value is returned …	if this occurs …
String containing a religion code	The value of the **religion** field was retrieved successfully.
Null	The **religion** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getRepatriationNumber

Description

The **getRepatriationNumber** method retrieves the value of a record's **repatriationNumber** field. Use this method to display the repatriation number for a person record returned from a demographic search.

Syntax

```
public java.lang.String getRepatriationNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getRepatriationNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a repatriation number	The value of the **repatriationNumber** field was retrieved successfully.
Null	The **repatriationNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getSpouseName

Description

The **getSpouseName** method retrieves the value of a record's **spouseName** field. Use this method to display the spouse's name for a person record returned from a demographic search.

Syntax

```
public java.lang.String getSpouseName()
```

Parameters

Parameter	Description
None	

Return Value

The **getSpouseName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **spouseName** field was retrieved successfully.
Null	The **spouseName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getSsn

Description

The **getSsn** method retrieves the value of the record's **ssn** field. Use this method to display the social security number in a person record returned from a demographic search.

Syntax

```
public Ssn getSsn()
```

Parameters

Parameter	Description
None	

Return Value

The **getSsn** method returns one of the following values:

This value is returned …	if this occurs …
Ssn object	The value of the **ssn** field was retrieved successfully.
Null	The **ssn** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getStatus

Description

The **getStatus** method retrieves an enumeration of the record's **status** field. Use this method to display the status of a person record returned from a demographic search.

Syntax

```
public EnumPersonStatus getStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getStatus** method returns one of the following values:

This value is returned …	if this occurs …
EnumPersonStatus object	The value of the **status** field was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString1

Description

The **getString1** method retrieves the value of a record's **string1** field. Use this method to display the value of the **string1** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString1()
```

Parameters

Parameter	Description
None	

Return Value

The **getString1** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string1** field was retrieved successfully.
Null	The **string1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString10

Description

The **getString10** method retrieves the value of a record's **string10** field. Use this method to display the value of the **string10** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString10()
```

Parameters

Parameter	Description
None	

Return Value

The **getString10** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string10** field was retrieved successfully.
Null	The **string10** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString2

Description

The **getString2** method retrieves the value of a record's **string2** field. Use this method to display the value of the **string2** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString2()
```

Parameters

Parameter	Description
None	

Return Value

The **getString2** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string2** field was retrieved successfully.
Null	The **string2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString3

Description

The **getString3** method retrieves the value of a record's **string3** field. Use this method to display the value of the **string3** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString3()
```

Parameters

Parameter	Description
None	

Return Value

The **getString3** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string3** field was retrieved successfully.
Null	The **string3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString4

Description

The **getString4** method retrieves the value of a record's **string4** field. Use this method to display the value of the **string4** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString4()
```

Parameters

Parameter	Description
None	

Return Value

The **getString4** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string4** field was retrieved successfully.
Null	The **string4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString5

Description

The **getString5** method retrieves the value of a record's **string5** field. Use this method to display the value of the **string5** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString5()
```

Parameters

Parameter	Description
None	

Return Value

The **getString5** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string5** field was retrieved successfully.
Null	The **string5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString6

Description

The **getString6** method retrieves the value of a record's **string6** field. Use this method to display the value of the **string6** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString6()
```

Parameters

Parameter	Description
None	

Return Value

The **getString6** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string6** field was retrieved successfully.
Null	The **string6** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString7

Description

The **getString7** method retrieves the value of a record's **string7** field. Use this method to display the value of the **string7** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString7()
```

Parameters

Parameter	Description
None	

Return Value

The **getString7** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string7** field was retrieved successfully.
Null	The **string7** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString8

Description

The **getString8** method retrieves the value of a record's **string8** field. Use this method to display the value of the **string8** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString8()
```

Parameters

Parameter	Description
None	

Return Value

The **getString8** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string8** field was retrieved successfully.
Null	The **string8** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getString9

Description

The **getString9** method retrieves the value of a record's **string9** field. Use this method to display the value of the **string9** field for a person record returned from a demographic search.

Syntax

```
public java.lang.String getString9()
```

Parameters

Parameter	Description
None	

Return Value

The **getString9** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string9** field was retrieved successfully.
Null	The **string9** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getSuffix

Description

The **getSuffix** method retrieves the value of a record's **suffix** field. Use this method to display the suffix for a person record returned from a demographic search.

Syntax

```
public java.lang.String getSuffix()
```

Parameters

Parameter	Description
None	

Return Value

The **getSuffix** method returns one of the following values:

This value is returned …	if this occurs …
String containing a suffix	The value of the **suffix** field was retrieved successfully.
Null	The **suffix** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getTitle

Description

The **getTitle** method retrieves the value of a record's **title** field. Use this method to display the title for a person record returned from a demographic search.

Syntax

```
public java.lang.String getTitle()
```

Parameters

Parameter	Description
None	

Return Value

The **getTitle** method returns one of the following values:

This value is returned …	if this occurs …
String containing a title	The value of the **title** field was retrieved successfully.
Null	The **title** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getUid

Description

The **getUid** method retrieves the value of the record's **uid** field. Use this method to display the UID of a person record returned from a demographic search.

Syntax

```
public Uid getUid()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid** method returns one of the following values:

This value is returned …	if this occurs …
UID object	The value of the **uid** field was retrieved successfully.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getVeteranStatus

Description

The **getVeteranStatus** method retrieves the value of a record's **veteranStatus** field. Use this method to display the veteran status for a person record returned from a demographic search.

Syntax

```
public java.lang.String getVeteranStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getVeteranStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing a status code	The value of the **veteranStatus** field was retrieved successfully.
Null	The **veteranStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

getVipFlag

Description

The **getVipFlag** method retrieves the value of a record's **vipFlag** field. Use this method to display the VIP status for a person record returned from a demographic search.

Syntax

```
public java.lang.String getVipFlag()
```

Parameters

Parameter	Description
None	

Return Value

The **getVipFlag** method returns one of the following values:

This value is returned …	if this occurs …
String containing a VIP status code	The value of the **vipStatus** field was retrieved successfully.
Null	The **vipStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsResultRO Class can be used, see the example for **getFirstName** beginning on page 4-208.

toString

Description

The **toString** method retrieves a comma-delimited representation of the instance.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
Comma-delimited list of demographic data	The specified demographic data for the record was retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in the **java.lang.Object** class (see your Java documentation for more information about this method)

Example

The following example performs an alphanumeric search, returning the resulting records in a PersonSearchResult object named **psr**. Each record in **psr** is a DemographicsResultRO object, represented by the variable **dr**. Once the search is performed, the example scrolls through each record in **psr**, and calls the **toString** method in the DemographicsResultRO class to retrieve and print a comma-delimited string for each **dr** object. The list below illustrates the result of calling **toString**.

```
1000000002,ELIZABETH,WARREN,null,12/12/1950,F,222112222,M,8.32
1000000003,ELIZABETH,WARREN,J,05/14/1960,F,777444777,M,8.32
1000000006,ELIZABETH,WARREN,M,05/14/1960,F,null,A,8.32
```

```
.../* Instantiating EiServer and creating
      SearchParameters object searchParams */

 PersonSearchResult psr = null;
 DemographicsResultRO dr = null;
 psr = personBO.searchAlpha(searchParams);
 SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

 if (psr==null) {
   System.out.println("No matching records.");
 } else {
   while(psr.next()) {
     dr = psr.getResultRow();
      System.out.println(dr.toString());
...
```

DemographicsRO Class

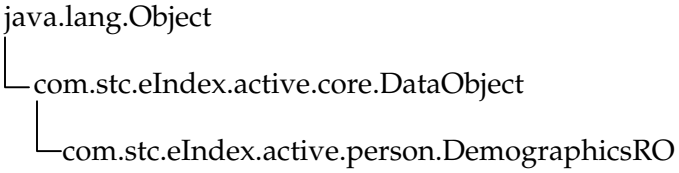
Description

The **DemographicsRO** class represents the demographic data in a person record. Field values for this class cannot be changed once initialized

Properties

The **DemographicsRO** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Methods

The methods included in the **DemographicsRO** class are described in detail on the following pages:

- **getString7** on page 4-285
- **getString8** on page 4-286
- **getString9** on page 4-287
- **getSuffix** on page 4-287
- **getTitle** on page 4-288
- **getVeteranStatus** on page 4-289
- **getVipFlag** on page 4-290

Inherited Methods

The **DemographicsRO** class inherits this method from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **DemographicsRO** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

getCitizenship

Description

The **getCitizenship** method retrieves the value of a record's **citizenship** field. Use this method to display a member's citizenship.

Syntax

```
public java.lang.String getCitizenship()
```

Parameters

Parameter	Description
None	

Return Value

The **getCitizenship** method returns one of the following values:

This value is returned …	if this occurs …
String containing a citizenship code	The value of the **citizenship** field was retrieved successfully.
Null	The **citizenship** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

The example on the following page performs an alphanumeric search based on predefined criteria, and then displays each field for the matching records. The example gets a PersonBO instance, and then initializes the variables **psr** (the search results object), **dr** (the DemographicsResultRO objects in **psr**), and **demo** (a DemographicsRO object). The example then performs the alphanumeric search and scrolls through the DemographicsResultRO objects in **psr**. For each record, the UID is obtained and values for each field are displayed. Typically, you would only display the fields for the selected DemographicsResultRO record rather than all records returned from a search.

```
.../* Instantiating EiServer and creating
      SearchParameters object searchParams */

PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 PersonSearchResult psr = null;
 DemographicsResultRO dr = null;
 DemographicsRO demo = null;
 psr = personBO.searchAlpha(searchParams);
 SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy");

 if (psr==null) {
   System.out.println("No matching records.");
 } else {
   while(psr.hasNext()) {
     dr = psr.next();
     Uid uid = dr.getUid();
     demo = personBO.getDemographics(uid);
       String dob = sdf.format(demo.getDob());
       System.out.println("First Name: " + demo.getFirstName());
       System.out.println("Last Name: " + demo.getLastName());
       System.out.println("Middle Name: " + demo.getMiddleName());
       System.out.println("Gender: " + get.Gender);
       System.out.println("SSN: " + demo.getSsn());
       System.out.println("Maiden Name: " + demo.getMaidenName());
       System.out.println("Mother Maiden: " + demo.getMotherMaidenName());
       System.out.println("Date of Birth: " + dob);
       System.out.println("City of Birth: " + demo.getPobCity());
       System.out.println("Status: " + demo.getStatus());
       System.out.println("MaritalStatus: " + demo.getMaritalStatus());
       System.out.println("Race: " + demo.getRace)));
       System.out.println("Suffix: " + demo.getSuffix());
       System.out.println("Title: " + demo.getTitle());
       System.out.println("String 1: " + demo.getString1());
       System.out.println("Class 2: " + demo.getClass2());
       System.out.println("LGA Code: " + demo.getLgaCode());
       System.out.println("Military Branch: " + demo.getMilitaryBranch());
...
```

getClass1

Description

The **getClass1** method retrieves the value of a record's **class1** field. Use this method to display the value of the **class1** field.

Syntax

```
public java.lang.String getClass1()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass1** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **class1** field was retrieved successfully.
Null	The **class1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getClass2

Description

The **getClass2** method retrieves the value of a record's **class2** field. Use this method to display the value of the **class2** field

Syntax

```
public java.lang.String getClass2()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass2** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **class2** field was retrieved successfully.
Null	The **class2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getClass3

Description

The **getClass3** method retrieves the value of a record's **class3** field. Use this method to display the value of the **class3** field.

Syntax

```
public java.lang.String getClass3()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass3** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **class3** field was retrieved successfully.
Null	The **class3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getClass4

Description

The **getClass4** method retrieves the value of a record's **class4** field. Use this method to display the value of the **class4** field.

Syntax

```
public java.lang.String getClass4()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass4** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **class4** field was retrieved successfully.
Null	The **class4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getClass5

Description

The **getClass5** method retrieves the value of a record's **class5** field. Use this method to display the value of the **class5** field.

Syntax

```
public java.lang.String getClass5()
```

Parameters

Parameter	Description
None	

Return Value

The **getClass5** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **class5** field was retrieved successfully.
Null	The **class5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDate1

Description

The **getDate1** method retrieves the value of a record's **date1** field. Use this method to display the value of the **date1** field.

Syntax

```
public java.sql.Date getDate1()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate1** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date1** field was retrieved successfully.
Null	The **date1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDate2

Description

The **getDate2** method retrieves the value of a record's **date2** field. Use this method to display the value of the **date2** field.

Syntax

```
public java.sql.Date getDate2()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate2** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date2** field was retrieved successfully.
Null	The **date2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDate3

Description

The **getDate3** method retrieves the value of a record's **date3** field. Use this method to display the value of the **date3** field.

Syntax

```
public java.sql.Date getDate3()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate3** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date3** field was retrieved successfully.
Null	The **date3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDate4

Description

The **getDate4** method retrieves the value of a record's **date4** field. Use this method to display the value of the **date4** field.

Syntax

```
public java.sql.Date getDate4()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate4** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date4** field was retrieved successfully.

This value is returned …	if this occurs …
Null	The **date4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDate5

Description

The **getDate5** method retrieves the value of a record's **date5** field. Use this method to display the value of the **date5** field.

Syntax

```
public java.sql.Date getDate5()
```

Parameters

Parameter	Description
None	

Return Value

The **getDate5** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **date5** field was retrieved successfully.
Null	The **date5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDateOfDeath

Description

The **getDateOfDeath** method retrieves the value of a record's **dateOfDeath** field. Use this method to display a member's date of death.

Syntax

```
public java.sql.Date getDateOfDeath()
```

Parameters

Parameter	Description
None	

Return Value

The **getDateOfDeath** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **dateOfDeath** field was retrieved successfully.
Null	The **dateOfDeath** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDeath

Description

The **getDeath** method retrieves the value of a record's **death** field. Use this method to display the value of the death indicator in a person record.

Syntax

```
public java.lang.String getDeath()
```

Parameters

Parameter	Description
None	

Return Value

The **getDeath** method returns one of the following values:

This value is returned …	if this occurs …
String containing a death indicator	The value of the **death** field was retrieved successfully.
Null	The **death** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDeathCertificate

Description

The **getDeathCertificate** method retrieves the value of a record's **deathCertificate** field. Use this method to display a member's death certificate number.

Syntax

```
public java.lang.String getDeathCertificate()
```

Parameters

Parameter	Description
None	

Return Value

The **getDeathCertificate** method returns one of the following values:

This value is returned …	if this occurs …
String containing death certificate information	The value of the **deathCertificate** field was retrieved successfully.
Null	The **deathCertificate** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDistrictOfResidence

Description

The **getDistrictOfResidence** method retrieves the value of a record's **districtOfResidence** field. Use this method to display a member's district of residence (DOR).

Syntax

```
public java.lang.String getDistrictOfResidence()
```

Parameters

Parameter	Description
None	

Return Value

The **getDistrictOfResidence** method returns one of the following values:

This value is returned …	if this occurs …
String containing a DOR code	The value of the **districtOfResidence** field was retrieved successfully.
Null	The **districtOfResidence** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDob

Description

The **getDob** method retrieves the value of a record's **dob** field. Use this method to display a member's date of birth.

Syntax

```
public java.sql.Date getDob()
```

Parameters

Parameter	Description
None	

Return Value

The **getDob** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **dob** field was retrieved successfully.
Null	The **dob** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDriversLicenseNumber

Description

The **getDriversLicenseNumber** method retrieves the value of a record's **driversLicenseNumber** field. Use this method to display a member's driver license number.

Syntax

```
public java.lang.String getDriversLicenseNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getDriversLicenseNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a driver license number	The value of the **driversLicenseNumber** field was retrieved successfully.
Null	The **driversLicenseNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getDriversLicenseState

Description

The **getDriversLicenseState** method retrieves the value of a record's **driversLicenseState** field. Use this method to display state that issued a member's driver license.

Syntax

```
public java.lang.String getDriversLicenseState()
```

Parameters

Parameter	Description
None	

Return Value

The **getDriversLicenseState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **driversLicenseState** field was retrieved successfully.
Null	The **driversLicenseState** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getEthnic

Description

The **getEthnic** method retrieves the value of a record's **ethnic** field. Use this method to display the ethnicity specified in a person record.

Syntax

```
public java.lang.String getEthnic()
```

Parameters

Parameter	Description
None	

Return Value

The **getEthnic** method returns one of the following values:

This value is returned …	if this occurs …
String containing an ethnic type	The value of the **ethnic** field was retrieved successfully.
Null	The **ethnic** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getFatherName

Description

The **getFatherName** method retrieves the value of a record's **fatherName** field. Use this method to display the father's name in a person record.

Syntax

```
public java.lang.String getFatherName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFatherName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **fatherName** field was retrieved successfully.
Null	The **fatherName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getFirstName

Description

The **getFirstName** method retrieves the value of a record's **firstName** field. Use this method to display the first name in a person record.

Syntax

```
public java.lang.String getFirstName()
```

Parameters

Parameter	Description
None	

Return Value

The **getFirstName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a first name	The value of the **firstName** field was retrieved successfully.
Null	The **firstName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getGender

Description

The **getGender** method retrieves the value of a record's **gender** field. Use this method to display the gender specified in a person record.

Syntax

```
public java.lang.String getGender()
```

Parameters

Parameter	Description
None	

Return Value

The **getGender** method returns one of the following values:

This value is returned …	if this occurs …
String containing a gender type	The value of the **gender** field was retrieved successfully.
Null	The **gender** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getLanguage

Description

The **getLanguage** method retrieves the value of a record's **language** field. Use this method to display the language specified in a person record.

Syntax

```
public java.lang.String getLanguage()
```

Parameters

Parameter	Description
None	

Return Value

The **getLanguage** method returns one of the following values:

This value is returned …	if this occurs …
String containing a language	The value of the **language** field was retrieved successfully.
Null	The **language** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getLastName

Description

The **getLastName** method retrieves the value of a record's **lastName** field. Use this method to display a last name in a person record.

Syntax

```
public java.lang.String getLastName()
```

Parameters

Parameter	Description
None	

Return Value

The **getLastName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a last name	The value of the **lastName** field was retrieved successfully.
Null	The **lastName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getLgaCode

Description

The **getLgaCode** method retrieves the value of a record's **lgaCode** field. Use this method to display a member's LGA code.

Syntax

```
public java.lang.String getLgaCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getLgaCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing an LGA code	The value of the **lgaCode** field was retrieved successfully.
Null	The **lgaCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMaidenName

Description

The **getMaidenName** method retrieves the value of a record's **maidenName** field. Use this method to display the maiden name in a person record.

Syntax

```
public java.lang.String getMaidenName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMaidenName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **maidenName** field was retrieved successfully.
Null	The **maidenName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMaritalStatus

Description

The **getMaritalStatus** method retrieves the value of a record's **maritalStatus** field. Use this method to display the marital status specified in a person record.

Syntax

```
public java.lang.String getMaritalStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getMaritalStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing a marital status	The value of the **maritalStatus** field was retrieved successfully.
Null	The **maritalStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMiddleName

Description

The **getMiddleName** method retrieves the value of a record's **middleName** field. Use this method to display the middle name or initial in a person record.

Syntax

```
public java.lang.String getMiddleName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMiddleName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a middle name or initial	The value of the **middleName** field was retrieved successfully.
Null	The **middleName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMilitaryBranch

Description

The **getMilitaryBranch** method retrieves the value of a record's **militaryBranch** field. Use this method to display the military branch specified in a person record.

Syntax

```
public java.lang.String getMilitaryBranch()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryBranch** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military branch	The value of the **militaryBranch** field was retrieved successfully.
Null	The **militaryBranch** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMilitaryRank

Description

The **getMilitaryRank** method retrieves the value of a record's **militaryRank** field. Use this method to display the military rank specified in a person record.

Syntax

```
public java.lang.String getMilitaryRank()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryRank** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military rank	The value of the **militaryRank** field was retrieved successfully.
Null	The **militaryRank** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMilitaryStatus

Description

The **getMilitaryStatus** method retrieves the value of a record's **militaryStatus** field. Use this method to display the military status specified in a person record.

Syntax

```
public java.lang.String getMilitaryStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getMilitaryStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing the name of a military status	The value of the **militaryStatus** field was retrieved successfully.
Null	The **militaryStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMotherMaidenName

Description

The **getMotherMaidenName** method retrieves the value of a record's **motherMaidenName** field. Use this method to display the mother's maiden name in a person record.

Syntax

```
public java.lang.String getMotherMaidenName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMotherMaidenName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a maiden name	The value of the **motherMaidenName** field was retrieved successfully.
Null	The **motherMaidenName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getMotherName

Description

The **getMotherName** method retrieves the value of a record's **motherName** field. Use this method to display the mother's name in a person record.

Syntax

```
public java.lang.String getMotherName()
```

Parameters

Parameter	Description
None	

Return Value

The **getMotherName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **motherName** field was retrieved successfully.
Null	The **motherName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getNationality

Description

The **getNationality** method retrieves the value of a record's **nationality** field. Use this method to display a member's nationality.

Syntax

```
public java.lang.String getNationality()
```

Parameters

Parameter	Description
None	

Return Value

The **getNationality** method returns one of the following values:

This value is returned …	if this occurs …
String containing a nationality code	The value of the **nationality** field was retrieved successfully.
Null	The **nationality** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPensionExpirationDate

Description

The **getPensionExpirationDate** method retrieves the value of a record's **pensionExpirationDate** field. Use this method to display the expiration date of a member's pension.

Syntax

```
public java.sql.Date getPensionExpirationDate()
```

Parameters

Parameter	Description
None	

Return Value

The **getPensionExpirationDate** method returns one of the following values:

This value is returned …	if this occurs …
Date	The value of the **pensionExpirationDate** field was retrieved successfully.
Null	The **pensionExpirationDate** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPensionNumber

Description

The **getPensionNumber** method retrieves the value of a record's **pensionNumber** field. Use this method to display a member's pension number.

Syntax

```
public java.lang.String getPensionNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getPensionNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a pension number	The value of the **pensionNumber** field was retrieved successfully.
Null	The **pensionNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPersonCategoryCode

Description

The **getPersonCategoryCode** method retrieves the value of a record's **personCategoryCode** field. Use this method to display the person category code assigned to a member.

Syntax

```
public java.lang.String getPersonCategoryCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getPersonCategoryCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a person category code	The value of the **personCategoryCode** field was retrieved successfully.
Null	The **personCategoryCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPobCity

Description

The **getPobCity** method retrieves the value of a record's **pobCity** field. Use this method to display the city of birth in a person record.

Syntax

```
public java.lang.String getPobCity()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobCity** method returns one of the following values:

This value is returned …	if this occurs …
String containing a city name	The value of the **pobCity** field was retrieved successfully.
Null	The **pobCity** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPobCountry

Description

The **getPobCountry** method retrieves the value of a record's **pobCountry** field. Use this method to display the country of birth in a person record.

Syntax

```
public java.lang.String getPobCountry()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobCountry** method returns one of the following values:

This value is returned …	if this occurs …
String containing a country code	The value of the **pobCountry** field was retrieved successfully.

This value is returned …	if this occurs …
Null	The **pobCountry** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getPobState

Description

The **getPobState** method retrieves the value of a record's **pobState** field. Use this method to display the state of birth in a person record.

Syntax

```
public java.lang.String getPobState()
```

Parameters

Parameter	Description
None	

Return Value

The **getPobState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **pobState** field was retrieved successfully.
Null	The **pobState** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getRace

Description

The **getRace** method retrieves the value of a record's **race** field. Use this method to display the race specified in a person record.

Syntax

```
public java.lang.String getRace()
```

Parameters

Parameter	Description
None	

Return Value

The **getRace** method returns one of the following values:

This value is returned …	if this occurs …
String containing a race code	The value of the **race** field was retrieved successfully.
Null	The **race** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getReligion

Description

The **getReligion** method retrieves the value of a record's **religion** field. Use this method to display the religion specified in a person record.

Syntax

```
public java.lang.String getReligion()
```

Parameters

Parameter	Description
None	

Return Value

The **getReligion** method returns one of the following values:

This value is returned …	if this occurs …
String containing a religion code	The value of the **religion** field was retrieved successfully.
Null	The **religion** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getRepatriationNumber

Description

The **getRepatriationNumber** method retrieves the value of a record's **repatriationNumber** field. Use this method to display a member's repatriation number.

Syntax

```
public java.lang.String getRepatriationNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getRepatriationNumber** method returns one of the following values:

This value is returned …	if this occurs …
String containing a repatriation number	The value of the **repatriationNumber** field was retrieved successfully.
Null	The **repatriationNumber** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getSpouseName

Description

The **getSpouseName** method retrieves the value of a record's **spouseName** field. Use this method to display the spouse name in a person record.

Syntax

```
public java.lang.String getSpouseName()
```

Parameters

Parameter	Description
None	

Return Value

The **getSpouseName** method returns one of the following values:

This value is returned …	if this occurs …
String containing a name	The value of the **spouseName** field was retrieved successfully.
Null	The **spouseName** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getSsn

Description

The **getSsn** method retrieves the value of a record's **ssn** field. Use this method to display the social security number associated with a person record.

Syntax

```
public Ssn getSsn()
```

Parameters

Parameter	Description
None	

Return Value

The **getSsn** method returns one of the following values:

This value is returned …	if this occurs …
Ssn object	The value of the **ssn** field was retrieved successfully.
Null	The **ssn** field was checked successfully but there was no value to retrieve.

Throws

None.

Additional Information

For more information about the **Ssn Class**, see page 4-475.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getStatus

Description

The **getStatus** method retrieves the value of a record's **status** field. Use this method to display the status of a person record.

Syntax

```
public EnumPersonStatus getStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getStatus** method returns one of the following values:

This value is returned …	if this occurs …
EnumPersonStatus object	The value of the **status** field was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString1

Description

The **getString1** method retrieves the value of a record's **string1** field. Use this method to display the value of the **string1** field.

Syntax

```
public java.lang.String getString1()
```

Parameters

Parameter	Description
None	

Return Value

The **getString1** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string1** field was retrieved successfully.
Null	The **string1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString10

Description

The **getString10** method retrieves the value of a record's **string10** field. Use this method to display the value of the **string10** field.

Syntax

```
public java.lang.String getString10()
```

Parameters

Parameter	Description
None	

Return Value

The **getString10** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string10** field was retrieved successfully.
Null	The **string10** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString2

Description

The **getString2** method retrieves the value of a record's **string2** field. Use this method to display the value of the **string2** field.

Syntax

```
public java.lang.String getString2()
```

Parameters

Parameter	Description
None	

Return Value

The **getString2** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string2** field was retrieved successfully.
Null	The **string2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString3

Description

The **getString3** method retrieves the value of a record's **string3** field. Use this method to display the value of the **string3** field.

Syntax

```
public java.lang.String getString3()
```

Parameters

Parameter	Description
None	

Return Value

The **getString3** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string3** field was retrieved successfully.
Null	The **string3** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString4

Description

The **getString4** method retrieves the value of a record's **string4** field. Use this method to display the value of the **string4** field.

Syntax

```
public java.lang.String getString4()
```

Parameters

Parameter	Description
None	

Return Value

The **getString4** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string4** field was retrieved successfully.
Null	The **string4** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString5

Description

The **getString5** method retrieves the value of a record's **string5** field. Use this method to display the value of the **string5** field.

Syntax

```
public java.lang.String getString5()
```

Parameters

Parameter	Description
None	

Return Value

The **getString5** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string5** field was retrieved successfully.
Null	The **string5** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString6

Description

The **getString6** method retrieves the value of a record's **string6** field. Use this method to display the value of the **string6** field.

Syntax

```
public java.lang.String getString6()
```

Parameters

Parameter	Description
None	

Return Value

The **getString6** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string6** field was retrieved successfully.
Null	The **string6** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString7

Description

The **getString7** method retrieves the value of a record's **string7** field. Use this method to display the value of the **string7** field.

Syntax

```
public java.lang.String getString7()
```

Parameters

Parameter	Description
None	

Return Value

The **getString7** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string7** field was retrieved successfully.
Null	The **string7** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString8

Description

The **getString8** method retrieves the value of a record's **string8** field. Use this method to display the value of the **string8** field.

Syntax

```
public java.lang.String getString8()
```

Parameters

Parameter	Description
None	

Return Value

The **getString8** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string8** field was retrieved successfully.
Null	The **string8** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getString9

Description

The **getString9** method retrieves the value of a record's **string9** field. Use this method to display the value of the **string9** field.

Syntax

```
public java.lang.String getString9()
```

Parameters

Parameter	Description
None	

Return Value

The **getString9** method returns one of the following values:

This value is returned …	if this occurs …
String	The value of the **string9** field was retrieved successfully.
Null	The **string9** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getSuffix

Description

The **getSuffix** method retrieves the value of a record's **suffix** field. Use this method to display the suffix specified in a person record.

Syntax

```
public java.lang.String getSuffix()
```

Parameters

Parameter	Description
None	

Return Value

The **getSuffix** method returns one of the following values:

This value is returned …	if this occurs …
String containing a suffix	The value of the **suffix** field was retrieved successfully.
Null	The **suffix** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getTitle

Description

The **getTitle** method retrieves the value of a record's **title** field. Use this method to display the title specified in a person record.

Syntax

```
public java.lang.String getTitle()
```

Parameters

Parameter	Description
None	

Return Value

The **getTitle** method returns one of the following values:

This value is returned …	if this occurs …
String containing a title	The value of the **title** field was retrieved successfully.
Null	The **title** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getVeteranStatus

Description

The **getVeteranStatus** method retrieves the value of a record's **veteranStatus** field. Use this method to display the veteran status in a person record.

Syntax

```
public java.lang.String getVeteranStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getVeteranStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing a veteran status	The value of the **veteranStatus** field was retrieved successfully.
Null	The **veteranStatus** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

getVipFlag

Description

The **getVipFlag** method retrieves the value of a record's **vipFlag** field. Use this method to display the VIP flag in a person record.

Syntax

```
public java.lang.String getVipFlag()
```

Parameters

Parameter	Description
None	

Return Value

The **getVipFlag** method returns one of the following values:

This value is returned …	if this occurs …
String containing a VIP status	The value of the **vipFlag** field was retrieved successfully.
Null	The **vipFlag** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how the '**get**' methods in the DemographicsRO class can be used, see the example for **getCitizenship** beginning on page 4-245.

EiBOFactory Class

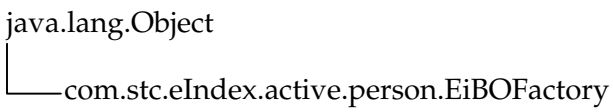
Description

The **EiBOFactory** class is the factory for EiSystem, Person, and LocalId business objects. A factory class is used to construct objects of these types in order to maintain singleton patterns for the objects.

Properties

The **EiBOFactory** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Methods

The methods included in the **EiBOFactory** class are described in detail on the following pages:

- **getEiSystemBOInstance** on page 4-292
- **getLocalIdBOInstance** on page 4-293
- **getPersonBOInstance** on page 4-294

Inherited Methods

The **EiBOFactory** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getEiSystemBOInstance

Description

The **getEiSystemBOInstance** method returns a single instance of the EiSystemBO class, maintaining a singleton pattern. The EiSystemBO class is thread-safe and stateless, so there is no need for multiple instances.

Syntax

```
public EiSystemBO getEiSystemBOInstance()
```

Parameters

Parameter	Description
None	

Return Value

The **getEiSystemBOInstance** method returns one of the following values:

This value is returned …	if this occurs …
Instance of EiSystemBO class	The instance of the EiSystemBO class was retrieved successfully.

Throws

None.

Example

The sample below retrieves a new instance of EiServer, and then calls **getEiSystemBOInstance** to retrieve a new instance of the EiSystemBO class.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 EiSystemBO systemBO =
     eiServer.getEiBOFactory().getEiSystemBOInstance();
...
```

getLocalIdBOInstance

Description

The **getLocalIdBOInstance** method returns a single instance of the LocalIdBO class, maintaining a singleton pattern. The LocalIdBO class is thread-safe and stateless, so there is no need for multiple instances.

Syntax

```
public LocalIdBO getLocalIdBOInstance()
```

Parameters

Parameter	Description
None	

Return Value

The **getLocalIdBOInstance** method returns one of the following values:

This value is returned …	if this occurs …
Instance of LocalIdBO class	The instance of the LocalIdBO class was retrieved successfully.

Throws

None.

Example

The sample below retrieves a new instance of EiServer, and then calls **getLocalIdBOInstance** to retrieve a new instance of the LocalIdBO class.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 LocalIdBO localIdBO = eiServer.getEiBOFactory().getLocalIdBOInstance();
...
```

getPersonBOInstance

Description

The **getPersonBOInstance** method returns a single instance of the PersonBO class, maintaining a singleton pattern. The PersonBO class it thread-safe and stateless, so there is no need for multiple instances.

Syntax

```
public PersonBO getPersonBOInstance()
```

Parameters

Parameter	Description
None	

Return Value

The **getPersonBOInstance** method returns one of the following values:

This value is returned …	if this occurs …
Instance of PersonBO class	The instance of the PersonBO class was retrieved successfully.

Throws

The **getPersonBOInstance** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Example

The sample below retrieves a new instance of EiServer, and then calls **getPersonBOInstance** to retrieve a new instance of the PersonBO class.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
...
```

EiConnection Class

Description

The **EiConnection** class wraps the standard JDBC connection class and provides services for statement caching and tracing.

Properties

The **EiConnection** class has the following properties:

- Public class
- Implements **java.sql.Connection**
- Extends **java.lang.Object**

java.lang.Object
└──com.stc.eIndex.active.person.EiConnection

Constructor

None.

Methods

The methods included in the **EiConnection** class are described in detail on the pages listed below. For more information about these methods, see your Java documentation for the **java.sql.Connection** interface.

- prepareStatement on page 4-309
- rollback on page 4-311
- setAutoCommit on page 4-312
- setCatalog on page 4-312
- setReadOnly on page 4-313
- setTransactionIsolation() on page 4-314
- setTypeMap on page 4-314

Inherited Methods

The **EiConnection** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

The **EiConnection** class inherits these methods from **java.sql.Connection** (see your Java documentation for more information):

- **transaction_none**
- **transaction_read_committed**
- **transaction_read_uncommitted**
- **transaction_repeatable_read**
- **transaction_serializable**

clearWarnings

Description

The **clearWarnings** method clears all warnings reported for the specified EiConnection object.

Syntax

```
public void clearWarnings()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

The **clearWarnings** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **clearWarnings** in interface **java.sql.Connection**

close

Description

The **close** method closes the logical connection to the database immediately.

Syntax

```
public void close()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

The **close** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **close** in interface **java.sql.Connection**

Example

The following example opens a new EiConnection **con**, and then calls **close** to close the connection.

```
  eiServer = new EiServer("EiServer.properties");
  con = eiServer.getConnection();
  ...

  con.close();
...
```

commit

Description

The **commit** method commits the current transaction to the database. It then releases the database locks, if any, held by the current connection. Only call this method when auto-commit is disabled.

Syntax

```
public void commit()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

The **commit** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **commit** in interface **java.sql.Connection**

createStatement

Description

The **createStatement** method creates a SQL statement object to use in a database query.

Syntax

```
public java.sql.Statement createStatement()
```

or

```
public java.sql.Statement createStatement(int p1, int p2)
```

Parameters

Parameter	Description
p1	A result set type (for more information, see ResultSet.TYPE_XXX in your Java documentation). This parameter is optional, but is required if you enter the **p2** parameter.
p2	A concurrency type (for more information, see ResultSet.CONCUR_XXX in your Java documentation). This parameter is optional, but is required if you enter the **p1** parameter.

Return Value

The **createStatement** method returns the following value:

This value is returned …	if this occurs …
A new statement object	The SQL statement object was created successfully.

Throws

The **createStatement** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **createStatement** in interface **java.sql.Connection**

getAutoCommit

Description

The **getAutoCommit** method checks whether or not auto-commit is enabled. By default, auto-commit is set to false.

Syntax

```
public boolean getAutoCommit()
```

Parameters

Parameter	Description
None	

Return Value

The **getAutoCommit** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	Auto-commit is currently enabled.
False	Auto-commit is currently disabled.

Throws

The **getAutoCommit** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getAutoCommit** in interface **java.sql.Connection**

getCatalog

Description

The **getCatalog** method retrieves the catalog name of the EiConnection object.

Syntax

```
public java.lang.String getCatalog()
```

Parameters

Parameter	Description
None	

Return Value

The **getCatalog** method returns one of the following values:

This value is returned …	if this occurs …
String containing a catalog name	The catalog name was retrieved successfully.
Null	There was no catalog name to retrieve.

Throws

The **getCatalog** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getCatalog** in interface **java.sql.Connection**

getEiServer

Description

The **getEiServer** method retrieves a handle to the EiServer object that created the current connection.

Syntax

```
public EiServer getEiServer()
```

Parameters

Parameter	Description
None	

Return Value

The **getEiServer** method returns one of the following values:

This value is returned …	if this occurs …
Handle to EiServer	The connection to the EiServer object was created successfully.

Throws

None.

getMetaData

Description

The **getMetaData** method retrieves metadata for the e*Index database, which supplies information about the database attributes.

Syntax

```
public java.sql.DatabaseMetaData getMetaData()
```

Parameters

Parameter	Description
None	

Return Value

The **getMetaData** method returns the following value:

This value is returned …	if this occurs …
DatabaseMetaData object	The metadata for the e*Index database was retrieved successfully.

Throws

The **getMetaData** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getMetaData** in interface **java.sql.Connection**

Example

The following example retrieves a new instance of the EiServer class, and then retrieves a new connection to the database. It then calls **getMetaData** and **getDatabaseProductName** to retrieve database information. Finally, the sample calls **prepareStatement** to select the database name.

```
...
 eiServer = new EiServer("EiServer.properties");
 con = eiServer.getConnection();
 if (con != null)  {
   System.out.println("Database Type: " +
                      con.getMetaData().getDatabaseProductName());
   PreparedStatement ps = con.prepareStatement("select name from v$database");
   ResultSet rs = ps.executeQuery();
...
```

getStartTransactionStatement

Description

The **getStartTransactionStatement** method returns a handle to the start transaction stored procedure. This method is used internally, and should not be used in your Java programs.

getTransactionIsolation

Description

The **getTransactionIsolation** method retrieves the transaction isolation level for the EiConnection object.

Syntax

```
public int getTransactionIsolation()
```

Parameters

Parameter	Description
None	

Return Value

The **getTransactionIsolation** method returns the following value:

This value is returned …	if this occurs …
Integer	The transaction isolation level was retrieved successfully.

Throws

The **getTransactionIsolation** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getTransactionIsolation** in interface **java.sql.Connection**

getTypeMap

Description

The **getTypeMap** method retrieves the type map for the given EiConnection object.

Syntax

```
public java.util.Map getTypeMap()
```

Parameters

Parameter	Description
None	

Return Value

The **getTypeMap** method returns the following value:

This value is returned …	if this occurs …
Map object	The type map for the EiConnection object was retrieved successfully.

Throws

The **getTypeMap** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getTypeMap** in interface **java.sql.Connection**

getWarnings

Description

The **getWarnings** method retrieves the first warning written by the current EiConnection.

Syntax

```
public java.sql.SQLWarning getWarnings()
```

Parameters

Parameter	Description
None	

Return Value

The **getWarnings** method returns one of the following values:

This value is returned …	if this occurs …
String containing a SQL warning	The first SQL warning for the current EiConnection object was retrieved successfully.
null	No SQL warnings have been written for the current EiConnection.

Throws

The **getWarnings** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **getWarnings** in interface **java.sql.Connection**

isClosed

Description

The **isClosed** method checks whether the connection is closed.

Syntax

```
public boolean isClosed()
```

Parameters

Parameter	Description
None	

Return Value

The **isClosed** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The given connection is closed.
False	The given connection is still open.

Throws

The **isClosed** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **isClosed** in interface **java.sql.Connection**

Example

The following example checks whether the EiConnection is closed, and prints a warning if the connection is closed.

```
...
  eiServer = new EiServer("EiServer.properties");
  con = eiServer.getConnection();
  ...

  boolean c = con.isClosed();
  if (c)
      System.out.println("WARNING!  Connection is closed.");
  else
      ...
```

isReadOnly

Description

The **isReadOnly** method checks whether the connection EiConnection is read-only.

Syntax

```
public boolean isReadOnly()
```

Parameters

Parameter	Description
None	

Return Value

The **isReadOnly** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The connection is currently read-only.
False	The connection is not currently read-only.

Throws

The **isReadOnly** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **isReadOnly** in interface **java.sql.Connection**

Example

The following example calls **isReadOnly** to check if the EiConnection **con** is in read-only mode. If **con** is read-only, it calls **setReadyOnly** to change the mode.

```
...
   con = eiServer.getConnection();
   boolean ro = con.isReadOnly();
   if (ro)
       con.setReadOnly(false);
   else
       ... /* Continue processing */
...
```

nativeSQL

Description

The **nativeSQL** method converts the specified SQL statement into native SQL.

Syntax

```
public java.lang.String nativeSQL(java.lang.String p1)
```

Parameters

Parameter	Description
p1	The SQL statement you want to convert to native SQL.

Return Value

The **nativeSQL** method returns one of the following values:

This value is returned …	if this occurs …
A SQL statement	The specified SQL statement was successfully converted into the system's native SQL language.

Throws

The **nativeSQL** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **nativeSQL** in interface **java.sql.Connection**

prepareCall

Description

The **prepareCall** method creates a CallableStatement object, which can then be used to execute stored procedures.

Syntax

```
public java.sql.CallableStatement prepareCall(java.lang.String p1)
```

or

```
public java.sql.CallableStatement prepareCall(java.lang.String p1, int p2, int p3)
```

Parameters

Parameter	Description
p1	A callable statement.
p2	A result set type (for more information, see Result.Set.TYPE_XXX in your Java documentation). This parameter is optional, but is required if the **p3** parameter is specified.
p3	A result concurrency type (for more information, see Result.Set.CONCUR_XXX in your Java documentation) . This parameter is optional, but is required if the **p2** parameter is specified.

Return Value

The **prepareCall** method returns the following value:

This value is returned …	if this occurs …
CallableStatement object	The SQL statement was created successfully.

Throws

The **prepareCall** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **prepareCall** in interface **java.sql.Connection**

prepareStatement

Description

The **prepareStatement** method creates a PreparedStatement object.

Syntax

```
public java.sql.PreparedStatement
prepareStatement(java.lang.String p1)
```

Note: ***prepareStatement*** *has two other possible forms. It can take either a PredefinedStatement object or a RuntimeStatement object. However these objects are only available internally and cannot be used in calls to* ***prepareStatement.***

Parameters

Parameter	Description
p1	The SQL statement you want to create.
resultSetType	A result set type (for more information, see ResultSet.TYPE_XXX in your Java documentation). This parameter is optional, but is required if a **resultSetConcurrency** is specified. These parameters can only be used with the **predefinedStatement** parameter.
resultSetConcurrency	A concurrency type (for more information, see ResultSet.CONCUR_XXX in your Java documentation). This parameter is optional, but is required if a **resultSetType** is specified.

Return Value

The **prepareStatement** method returns the following value:

This value is returned …	if this occurs …
PreparedStatement object	The SQL statement was created successfully.

Throws

The **prepareStatement** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **prepareStatement** in interface **java.sql.Connection**

Example

The following sample retrieves a new instance of the EiServer class, and then retrieves a new connection to the database. It then calls **getMetaData** and **getDatabaseProductName** to retrieve database information. Finally, the sample calls **prepareStatement** to select the database name and **executeQuery** to run the SQL statement against the database.

```
...
 eiServer = new EiServer("EiServer.properties");
 con = eiServer.getConnection();
 if (con != null)  {
   System.out.println("Database Type: " +
                      con.getMetaData().getDatabaseProductName());
   PreparedStatement ps = con.prepareStatement("select name from v$database");
   ResultSet rs = ps.executeQuery();
   if (rs.next()) {
       System.out.println("Database Name: " + rs.getString(1));
   } else {
       throw new EiException("Error fetching selection.");
...
```

rollback

Description

The **rollback** method rolls back the current transaction, reversing any changes to the database. This method also releases database locks, if any, held by the current connection. Only call this method if auto-commit is not enabled.

Syntax

```
public void rollback()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

The **rollback** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **rollback** in interface **java.sql.Connection**

setAutoCommit

Description

The **setAutoCommit** method specifies whether auto-commit is enabled. By default, auto-commit is set to false.

Syntax

```
public void setAutoCommit(boolean p1)
```

Parameters

Parameter	Description
p1	Either Boolean true or false. Entering a value of **True** enables auto-commit. Entering a value of **False** disables auto-commit.

Return Value

None.

Throws

The **setAutoCommit** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **setAutoCommit** in interface **java.sql.Connection**

setCatalog

Description

The **setCatalog** method is the setter method for the catalog name for the EiConnection object.

Syntax

```
public void setCatalog(java.lang.String p1)
```

Parameters

Parameter	Description
p1	The name for the connection's catalog.

Return Value

None.

Throws

The **setCatalog** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **setCatalog** in interface **java.sql.Connection**

setReadOnly

Description

The **setReadOnly** method specifies that the connection is read-only.

Syntax

```
public void setReadOnly(boolean p1)
```

Parameters

Parameter	Description
p1	Boolean true or false. Entering a value of **True** turns on read-only mode; entering a value of **False** turns off read-only mode.

Return Value

None.

Throws

The **setReadOnly** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **setReadOnly** in interface **java.sql.Connection**

Example

The following example calls **isReadOnly** to check if the EiConnection **con** is in read-only mode. If **con** is read-only, it calls **setReadyOnly** to change the mode.

```
...
   con = eiServer.getConnection();
   boolean ro = con.isReadOnly();
   if (ro)
       con.setReadOnly(false);
   else
       ... /* Continue processing */
...
```

setTransactionIsolation()

Description

The **setTransactionIsolation** method changes the transaction isolation level to the specified level.

Syntax

```
public void setTransactionIsolation(int p1)
```

Parameters

Parameter	Description
p1	An integer representing the transaction isolation level.

Return Value

None.

Throws

The **setTransactionIsolation** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **setTransactionIsolation** in interface **java.sql.Connection**

setTypeMap

Description

The **setTypeMap** method specifies the type map for the connection.

Syntax

```
public void setTypeMap(java.util.Map p1)
```

Parameters

Parameter	Description
p1	The Map object to use as this EiConnection object's default type map.

Return Value

None.

Throws

The **setTypeMap** method throws the following exception:

- **java.sql.SQLException**

Additional Information

Specified By:

- **setTypeMap** in interface **java.sql.Connection**

EiNull Interface

Description

The **EiNull** interface allows you set fields to null and override existing values with null values during a person update. The fields in this class specify the data type of the field being set to null.

Properties

The **EiNull** interface has the following properties:

- Public interface
- interface com.stc.eIndex.active.person.EiNull

Constructor

None.

Fields

The fields included in the **EiNull** interface are described in detail on the following pages:

- **BOOLEAN** on page 4-317
- **FLOAT** on page 4-317
- **LONG** on page 4-317
- **SQL_DATE** on page 4-317
- **STRING** on page 4-318
- **TIMESTAMP** on page 4-318

Methods

None.

Inherited Methods

None.

BOOLEAN

Description

The **BOOLEAN** field specifies that the field you are setting to null is of the Boolean data type.

Syntax

```
public static final java.lang.Boolean BOOLEAN
```

FLOAT

Description

The **FLOAT** field specifies that the field you are setting to null is of the Float data type.

Syntax

```
public static final java.lang.Float FLOAT
```

LONG

Description

The **LONG** field specifies that the field you are setting to null is of the Long data type.

Syntax

```
public static final java.lang.Long LONG
```

SQL_DATE

Description

The **SQL_DATE** field specifies that the field you are setting to null is of the sql.date data type.

Syntax

```
public static final java.sql.Date DATE
```

STRING

Description

The **STRING** field specifies that the field you are setting to null is of the String data type.

Syntax

```
public static final java.lang.String STRING
```

TIMESTAMP

Description

The **TIMESTAMP** field specifies that the field you are setting to null is of the Timestamp data type.

Syntax

```
public static final java.sql.Timestamp TIMESTAMP
```

EiServer Class

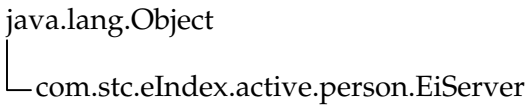
Description

The **EiServer** class is a container class for all singleton classes in the API. The constructor method for this class can either read a properties file for configuration and runtime information, or it can take the information from a Properties object that has been loaded with the appropriate information. The properties file for the sample code is named **EiServer.properties**. The path to the properties file you use must be defined in the **CLASSPATH** environment variable. If you use a Properties object instead of the properties file, this path does not need to be defined. The EiServer class must be instantiated before any other operation using the Active Integration APIs takes place.

Properties

The **EiServer** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

The **EiServer** class has one constructor, which is described on the following page:

- **EiServer** on page 4-321

Methods

The methods included in the **EiServer** class are described in detail on the following pages:

- **equals** on page 4-322
- **getConnection** on page 4-323
- **getDbTimestamp** on page 4-323
- **getEiBOFactory** on page 4-324
- **getLogicalName** on page 4-325
- **getProperties** on page 4-326
- **hashCode** on page 4-327

Inherited Methods

The **EiServer** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiServer

Description

The **EiServer** constructor method instantiates a new EiServer object based on the configuration of the specified property file or based on the properties information contained in a Properties object.

Syntax

```
public EiServer(java.lang.String propertyFileName)
```

or

```
public EiServer(java.util.Properties properties)
```

Parameters

Parameter	Description
propertyFileName	The name of the property file to read for configuration and runtime information.
properties	The name of the Properties object that contains the properties information for the current instance of EiServer.

Return Value

The **EiServer** method returns one of the following values:

This value is returned …	if this occurs …
Instance of EiServer class	The new EiServer object is instantiated successfully.
Error message	The specified property file is not found.

Throws

None.

Example

The following sample calls **EiServer** to retrieve a new instance of the EiServer class based on the property file named **EiServer.properties**. You must call **EiServer** to instantiate the EiServer class before calling any other e*Index methods. The sample then calls **getConnection** to obtain a connection to the database.

```
EiServer eiServer = new EiServer("EiServer.properties")
con = eiServer.getConnection();
```

equals

Description

The **equals** method compares one object with another to see whether any fields are different.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The object to be compared with the current object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The objects were compared successfully and all fields are identical between the two objects.
False	The objects were compared successfully and the fields are not identical between the two objects.

Throws

None.

Additional Information

Overrides:

- **equals** in the **java.lang.Object** class (see your Java documentation for more information about this method)

Example

The following example obtains an instance of EiServer named **eiServer**. It then obtains a new instance of EiServer, **newServer**, by calling **lookupEiServer** to find the instance of EiServer connected to the database whose logical name is **E451**. The example calls **equals** to check if the two objects are the same.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 EiServer newServer = eiServer.lookupEiServer("E451");
 if (eiServer.equals(newServer)) {
...
```

getConnection

Description

The **getConnection** method returns a connection to the e*Index database from the connection pool.

Syntax

```
public EiConnection  getConnection()
```

Parameters

Parameter	Description
None	

Return Value

The **getConnection** method returns the following value:

This value is returned …	if this occurs …
Connection to the database	The connection to the database is made successfully.

Throws

The **getConnection** method throws the following exception:

- **java.sql.SQLException**

Example

The following sample calls **EiServer** to retrieve a new instance of the EiServer class based on the property file named **EiServer.properties**. The sample then calls **getConnection** to obtain a connection to the database.

```
EiServer eiServer = new EiServer("EiServer.properties")
con = eiServer.getConnection();
```

getDbTimestamp

Description

The **getDbTimestamp** method retrieves the current date and time from the e*Index database.

Syntax

```
public java.sql.Timestamp getDbTimestamp()
```

Parameters

Parameter	Description
None	

Return Value

The **getDbTimestamp** method returns the following value:

This value is returned …	if this occurs …
Timestamp	The database timestamp was retrieved successfully.

Throws

None.

Example

The following example illustrates how to use the **getDbTimestamp** when creating an Audit object. The '**set**' methods are called to fill the Audit object with audit information, and **getDbTimestamp** is called within the call to **setTimestamp** in order to create a timestamp for the Audit object.

```
...
      Audit audit = new Audit();
        audit.setUserId(user_id);
        audit.setFunct(function);
        audit.setDetail(function_description);
        audit.setTimestamp(eiServer.getDbTimestamp());
...
```

getEiBOFactory

Description

The **getEiBOFactory** method retrieves the business object factory for the current instance of EiServer.

Syntax

```
public EiBOFactory getEiBOFactory()
```

Parameters

Parameter	Description
None	

Return Value

The **getEiBOFactory** method returns the following value:

This value is returned …	if this occurs …
EiBOFactory object	The business object factory is retrieved successfully.

Throws

None.

Example

The following sample calls **EiServer** to retrieve a new instance of the EiServer class based on the property file named **EiServer.properties**. The sample then calls **getEiBOFactory** to retrieve a factory object so a PersonBO object instance can be retrieved.

```
EiServer eiServer = new EiServer("EiServer.properties");
PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
```

getLogicalName

Description

The **getLogicalName** method retrieves the logical name specified in the properties file being used for the current instance of EiServer. Each instance of EiServer must have a unique logical name.

Syntax

```
public java.lang.String getLogicalName()
```

Parameters

Parameter	Description
None	

Return Value

The **getLogicalName** method returns the following value:

This value is returned …	if this occurs …
String containing a logical name	The logical name was retrieved successfully.

Throws

None.

Example

The following example obtains a new instance of EiServer named **eiServer**, and then calls **getLogicalName** to retrieve the logical name defined for **eiServer**.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 String name = eiServer.getLogicalName());
...
```

getProperties

Description

The **getProperties** method retrieves the property information from the Java properties file for the current instance of EiServer.

Syntax

```
public java.util.Properties getProperties()
```

Parameters

Parameter	Description
None	

Return Value

The **getProperties** method returns the following value:

This value is returned …	if this occurs …
String	The information from the properties file currently in use was retrieved successfully.

Throws

None.

Example

The following example obtains a new instance of EiServer, and then calls **getProperties** to display a string listing the parameters and parameter values from the properties file currently in use. Each parameter and value pair is delimited by a comma.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 System.out.println(newServer.getProperties());
...
```

hashCode

Description

The **hashCode** method returns a hash code value for the specified object.

Syntax

```
public int hashCode()
```

Parameters

Parameter	Description
None	

Return Value

The **hashCode** method returns the following value:

This value is returned …	if this occurs …
Integer	The hash code value was retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **hashCode** in the **java.lang.Object** class (see your Java documentation for more information about this method)

EiSystem Class

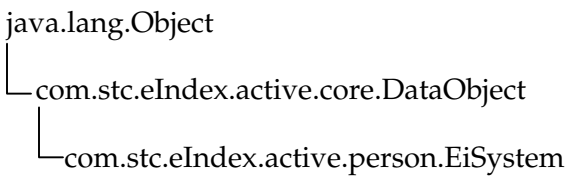
Description

The **EiSystem** class represents a system object that contains information about a specific system. This information is stored in the database in the *ui_facility* table.

Properties

The **EiSystem** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.core.DataObject**



Constructor

None.

Methods

The methods included in the **EiSystem** class are described in detail on the following pages:

- **getZip** on page 4-342
- **getZipExt** on page 4-343
- **isFormatMatch** on page 4-344
- **isLocalIdMaskEnabled** on page 4-345
- **isSystemMaskEnabled** on page 4-346

Inherited Methods

The **EiSystem** class inherits this method from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **EiSystem** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

getAddress1

Description

The **getAddress1** method retrieves the value of a system's **address1** field. Use this method to display the first line of a street address in a system record.

Syntax

```
public java.lang.String getAddress1()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress1** method returns one of the following values:

This value is returned …	if this occurs …
String containing a street address	The value of the **address1** field was retrieved successfully.
Null	The **address1** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

The following example retrieves a new instance of EiServer and a new instance of the EiSystemBO class. It then calls **getSystem** to retrieve the information associated with the system code **SBYN**. The example then displays each field associated with system **SBYN**.

```
EiServer eiServer = new EiServer("EiServer.properties");
EiSystemBO systemBO = eiServer.getEiBOFactory().getEiSystemBOInstance();
EiSystem sys = systemBO.getSystem("SBYN");
System.out.println("SYSTEM CODE:  " + sys.getSystemCode());
System.out.println("SYSTEM DESCRIPTION:  " + sys.getDescription());
System.out.println("REGION CODE:  " + sys.getRegionCode());
System.out.println("ADDRESS 1:  " + sys.getAddress1());
System.out.println("ADDRESS 2:  " + sys.getAddress2());
System.out.println("CITY:  " + sys.getCity());
System.out.println("STATE:  " + sys.getState());
System.out.println("ZIP CODE:  " + sys.getZip() + "-" + sys.getZipExt());
System.out.println("COUNTY:  " + sys.getCounty());
System.out.println("COUNTRY:  " + sys.getCountry());
System.out.println("LOCAL ID LENGTH:  " + sys.getIdLength());
System.out.println("ALLOW SHORTER ID?:  " + sys.getAllowLessIdLength());
System.out.println("LOCAL ID FORMAT:  " + sys.getLocalIdFormat());
System.out.println("NEXT LOCAL ID:  " + sys.getNextLocalId());
System.out.println("SYSTEM STATUS:  " + sys.getStatus());
System.out.println("MASK SYSTEM NAME?:  " + sys.isSystemMaskEnabled());
System.out.println("MASK LOCAL ID?:  " + sys.isLocalIdMaskEnabled());
System.out.println("CREATE DATE:  " + sys.getCreateDate());
```

getAddress2

Description

The **getAddress2** method retrieves the value of a system's **address2** field. Use this method to display the second line of a street address in a system record.

Syntax

```
public java.lang.String getAddress1()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddress2** method returns one of the following values:

This value is returned …	if this occurs …
String containing a street address	The value of the **address2** field was retrieved successfully.

This value is returned …	if this occurs …
Null	The **address2** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getAddress2** can be used, see the example for **getAddress1** beginning on page 4-330.

getAllowLessIdLength

Description

The **getAllowLessIdLength** method returns a Boolean value indicating whether you can enter a local ID for the specified system that has fewer characters than indicated in the **idLength** field.

Syntax

```
public boolean getAllowLessIdLength()
```

Parameters

Parameter	Description
None	

Return Value

The **getAllowLessIdLength** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The system allows local IDs with fewer characters than specified by the **idLength** field.
False	The system does not allow local IDs with fewer characters than specified by the **idLength** field.

Throws

None.

Example

To see an example of how **getAllowLessIdLength** can be used, see the example for **getAddress1** beginning on page 4-330.

getCity

Description

The **getCity** method retrieves the value of a system's **city** field. Use this method to display the city in which a system is located.

Syntax

```
public java.lang.String getCity()
```

Parameters

Parameter	Description
None	

Return Value

The **getCity** method returns one of the following values:

This value is returned …	if this occurs …
String containing a city name	The value of the **city** field was retrieved successfully.
Null	The **city** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getCity** can be used, see the example for **getAddress1** beginning on page 4-330.

getCountry

Description

The **getCountry** method retrieves the value of a system's **country** field. Use this method to display the country in which a system is located.

Syntax

```
public java.lang.String getCountry()
```

Parameters

Parameter	Description
None	

Return Value

The **getCountry** method returns one of the following values:

This value is returned …	if this occurs …
String containing a country name	The value of the **country** field was retrieved successfully.
Null	The **country** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getCountry** can be used, see the example for **getAddress1** beginning on page 4-330.

getCounty

Description

The **getCounty** method retrieves the value of a system's **county** field. Use this method to display the county in which a system is located.

Syntax

```
public java.lang.String getCounty()
```

Parameters

Parameter	Description
None	

Return Value

The **getCounty** method returns one of the following values:

This value is returned …	if this occurs …
String containing a county name	The value of the **county** field was retrieved successfully.
Null	The **county** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getCounty** can be used, see the example for **getAddress1** beginning on page 4-330.

getCreateDate

Description

The **getCreateDate** method retrieves the value of a system's **createDate** field. Use this method to display the date on which a system record was created or was last updated.

Syntax

```
public java.sql.Timestamp getCreateDate()
```

Parameters

Parameter	Description
None	

Return Value

The **getCreateDate** method returns one of the following values:

This value is returned …	if this occurs …
Timestamp	The value of the **createDate** field was retrieved successfully.
Null	There was no create date to retrieve.

Throws

None.

Example

To see an example of how **getCreateDate** can be used, see the example for **getAddress1** beginning on page 4-330.

getDescription

Description

The **getDescription** method retrieves the value of a system's **description** field. Use this method to display a description of a system.

Syntax

```
public java.lang.String getDescription()
```

Parameters

Parameter	Description
None	

Return Value

The **getDescription** method returns one of the following values:

This value is returned …	if this occurs …
String containing a system description	The value of the **description** field was retrieved successfully.
Null	The **description** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getDescription** can be used, see the example for **getAddress1** beginning on page 4-330.

getIdLength

Description

The **getIdLength** method retrieves the value of a system's **idLength** field. Use this method to display the preferred length for local ID's in a specific system.

Syntax

```
public int getIdLength()
```

Parameters

Parameter	Description
None	

Return Value

The **getIdLength** method returns one of the following values:

This value is returned …	if this occurs …
Integer	The value of the **idLength** field was retrieved successfully.
Null	The **idLength** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getDescription** can be used, see the example for **getAddress1** beginning on page 4-330.

getLocalIdFormat

Description

The **getLocalIdFormat** method retrieves the value of a system's **localIdFormat** field. Use this method to display the format in which local ID's for a system are displayed. Local ID formats are expressed by a series of characters: **!** (allows upper case alphanumeric characters), **@** (allows numeric characters), **#** (allows numeric characters), **A** (allows alphanumeric characters), or **X** (allows any character type).

Syntax

```
public java.lang.String getLocalIdFormat()
```

Parameters

Parameter	Description
None	

Return Value

The **getLocalIdFormat** method returns one of the following values:

This value is returned …	if this occurs …
String containing a series of characters	The value of the **localIdFormat** field was retrieved successfully.
Null	No format is specified for local IDs in the given system.

Throws

None.

Example

To see an example of how **getLocalIdFormat** can be used, see the example for **getAddress1** beginning on page 4-330.

getNextLocalId

Description

The **getNextLocalId** method retrieves the value of a system's **nextLocalId** field. Use this method to display the next local ID specified for a system, if any.

Note: *The next local ID is specified in the database table* ui_local_id_generator. *This table is described in chapter 3 of this guide.*

Syntax

```
public java.lang.String getNextLocalId()
```

Parameters

Parameter	Description
None	

Return Value

The **getNextLocalId** method returns one of the following values:

This value is returned …	if this occurs …
String containing a local ID	The value of the **nextLocalId** field was retrieved successfully.
Null	No next local ID was specified.

Throws

None.

Example

To see an example of how **getNextLocalId** can be used, see the example for **getAddress1** beginning on page 4-330.

getRegionCode

Description

The **getRegionCode** method retrieves the value of a system's **regionCode** field. Use this method to display the processing code for the region in which a system is located.

Syntax

```
public java.lang.String getRegionCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getRegionCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a region code	The value of the **regionCode** field was retrieved successfully.
Null	The **regionCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getRegionCode** can be used, see the example for **getAddress1** beginning on page 4-330.

getState

Description

The **getState** method retrieves the value of a system's **state** field. Use this method to display the state in which a system is located.

Syntax

```
public java.lang.String getState()
```

Parameters

Parameter	Description
None	

Return Value

The **getState** method returns one of the following values:

This value is returned …	if this occurs …
String containing a state code	The value of the **state** field was retrieved successfully.
Null	The **state** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getState** can be used, see the example for **getAddress1** beginning on page 4-330.

getStatus

Description

The **getStatus** method retrieves the value of a system's **status** field. Use this method to display a system's status.

Syntax

```
public java.lang.String getStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getStatus** method returns one of the following values:

This value is returned …	if this occurs …
String containing a status code	The value of the **status** field was retrieved successfully.
Null	The **status** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getStatus** can be used, see the example for **getAddress1** beginning on page 4-330.

getSystemCode

Description

The **getSystemCode** method retrieves the value of a system's **systemCode** field. Use this method to display the processing code of a system.

Syntax

```
public java.lang.String getSystemCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSystemCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a system code	The value of the **systemCode** field was retrieved successfully.
Null	The **systemCode** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getSystemCode** can be used, see the example for **getAddress1** beginning on page 4-330.

getZip

Description

The **getZip** method retrieves the value of a system's **zip** field. Use this method to display the zip code for a system's address.

Syntax

```
public java.lang.String getZip()
```

Parameters

Parameter	Description
None	

Return Value

The **getZip** method returns one of the following values:

This value is returned …	if this occurs …
String containing a zip code	The value of the **zip** field was retrieved successfully.
Null	The **zip** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getZip** can be used, see the example for **getAddress1** beginning on page 4-330.

getZipExt

Description

The **getZipExt** method retrieves the value of a system's **zipExt** field. Use this method to display the zip code extension of a system's address.

Syntax

```
public java.lang.String getZipExt()
```

Parameters

Parameter	Description
None	

Return Value

The **getZipExt** method returns one of the following values:

This value is returned …	if this occurs …
String containing a zip code extension	The value of the **zipExt** field was retrieved successfully.
Null	The **zipExt** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see an example of how **getZipExt** can be used, see the example for **getAddress1** beginning on page 4-330.

isFormatMatch

Description

The **isFormatMatch** method retrieves a Boolean true or false, indicating whether the format of a local ID matches the required format for the associated system.

Syntax

```
public boolean isFormatMatch(java.lang.String id, boolean
matchLessLength)
```

Parameters

Parameter	Description
id	The local ID whose formatting is being validated.
matchLessLength	A Boolean indicator of whether local IDs shorter than the length specified for the System object are allowed.

Return Value

The **isFormatMatch** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The specified local ID matches the format defined for the System object.
False	The specified local ID does not match the format defined for the System object.

Throws

None.

Example

The following example calls the **LocalId** constructor method to create a new LocalId object based on the system and local ID specified. It then calls **getAllowLessIdLength** to check if local IDs of variable lengths are allowed, and calls **isFormatMatch** to compare the given local ID with the required format. If the local ID passes the validation, a new local ID record is added. If it fails, an error message is displayed.

```
...
 LocalId lid = new LocalId(system_code,local_id);
 EiSystem sys = systemBO.getSystem(system_code);
 boolean length = sys.getAllowLessIdLength();
 if(!sys.isFormatMatch(local_id, length)) {
     System.out.println("WRONG FORMAT FOR LOCAL ID.");
 } else {
     personBO.addLocalId(person, lid);
 }
...
```

isLocalIdMaskEnabled

Description

The **isLocalIdMaskEnabled** method retrieves a Boolean true or false, indicating whether the local ID mask function is enabled for a system.

Note: *For more information about local ID masking, see chapter 4 of the* e*Index Administrator User's Guide.

Syntax

```
public boolean isLocalIdMaskEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isLocalIdMaskEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	Local ID masking is enabled for the specified system.
False	Local ID masking is not enabled for the specified system.

Throws

None.

Example

The following example retrieves new instances of EiServer, PersonBO, and EiSystemBO. It then loads a Person object with person information from the e*Index database, and creates an enumeration of the associated LocalId objects. For each local ID record in the enumeration, the example calls **getSystem** to create an EiSystem object containing information about the associated system. It then calls **isSystemMaskEnabled** and **isLocalIdMaskEnabled** to determine whether to display system and local ID information.

```
EiServer eiServer = new EiServer("EiServer.properties");
PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
EiSystemBO eiSys = eiServer.getEiBOFactory().getEiSystemBOInstance();

Uid uid = new Uid("1000000001");
Person person = new Person();
personBO.loadPerson(uid, person);
Enumeration e = person.getLocalIdEnumeration();
while (e.hasMoreElements()) {
    LocalId id = (LocalId)e.nextElement();
    String code = id.getSystemCode();
    if (code.equals("OLD #")) {
        System.out.println("MERGED UID:  " + id.getLocalId());
    }else {
        EiSystem system = eiSys.getSystem(code);
        if (system.isSystemMaskEnabled()) {
            System.out.println("**********");
        }else {
            System.out.println("SYSTEM:  " + system.getSystemCode());
        }
        if (system.isLocalIdMaskEnabled()) {
            System.out.println("**********");
        }else {
            System.out.println("LOCAL ID:  " + id.getLocalId());
        }
    }
}
```

isSystemMaskEnabled

Description

The **isSystemMaskEnabled** method returns a Boolean value that indicates whether the system name is hidden for the specified system.

Note: *For more information about system name masking, see chapter 4 of the* e*Index Administrator User's Guide.

Syntax

```
public boolean isSystemMaskEnabled()
```

Parameters

Parameter	Description
None	

Return Value

The **isSystemMaskEnabled** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	System name masking is enabled for the specified system.
False	System name masking is not enabled for the specified system.

Throws

None.

Example

To see an example of how **isSystemMaskEnabled** can be used, see the example for **isLocalIdMaskEnabled** on page 4-345.

EiSystemBO Class

Description

The **EiSystemBO** class is the business object class for the **EiSystem** class. This class contains methods that you use to retrieve information from the *ui_facility* table in the database.

Properties

The **EiSystemBO** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└─com.stc.eIndex.active.person.EiSystemBO

Constructor

None.

Methods

The methods included in the **EiSystemBO** class are described in detail on the following pages:

- **getSystem** on page 4-349
- **getSystemDescription** on page 4-350
- **getSystemEnumeration** on page 4-351

Inherited Methods

The **EiSystemBO** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getSystem

Description

The **getSystem** method returns an EiSystem object from the database given the system code.

Syntax

```
public EiSystem getSystem(java.lang.String systemCode)
```

Parameters

Parameter	Description
systemCode	The processing code for the system whose information you want to retrieve.

Return Value

The **getSystem** method returns one of the following values:

This value is returned …	if this occurs …
EiSystem object	The system information was retrieved successfully.
null	The specified system code does not exist in the *ui_facility* table.

Throws

None.

Example

The following example retrieves a new instance of EiServer and a new instance of the EiSystemBO class. It then calls **getSystem** to retrieve the information associated with the system code specified by the variable *system_code*. The system fields are delimited by commas, as shown below.

```
CBC,CAPE BURR CENTER,A,1259 SHORELINE DRIVE,null,CAPE
BURR,CT,09876,null,CAPE BURR,USA,SOUTH CENTRAL,2002-05-01
17:23:05.0,11,false,###-###-###-##,false,null,true
```

```
EiServer eiServer = new EiServer("EiServer.properties");
EiSystemBO system =
    eiServer.getEiBOFactory().getEiSystemBOInstance();
System.out.println(systemBO.getSystem(system_code));
```

getSystemDescription

Description

The **getSystemDescription** method retrieves the system description associated with the specified system code.

Syntax

```
public java.lang.String getSystemDescription(java.lang.String
systemCode)
```

Parameters

Parameter	Description
systemCode	The processing code for the system whose description you want to retrieve.

Return Value

The **getSystemDescription** method returns one of the following values:

This value is returned …	if this occurs …
String containing a system description	The system description was retrieved successfully.
null	The specified system code does not exist in the *ui_facility* table.

Throws

The **getSystemDescription** method throws the following exception:

- **EiException Class**

Example

The following example retrieves a new instance of EiServer and a new instance of the EiSystemBO class. It then calls **getSystemDescription** to retrieve the description associated with the system code **SBYN**. In this case, the return value is **SeeBeyond**.

```
EiServer eiServer = new EiServer("EiServer.properties");
EiSystemBO system =
    eiServer.getEiBOFactory().getEiSystemBOInstance();
System.out.println(systemBO.getSystemDescription("SBYN"));
```

getSystemEnumeration

Description

The **getSystemEnumeration** method retrieves a list of all systems that are stored in the *ui_facility* table.

Syntax

```
public java.util.Enumeration getSystemEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getSystemEnumeration** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration of systems	The system information was retrieved successfully.
Empty Enumeration	There was no system information to retrieve.

Throws

None.

Example

The following example retrieves a new instance of EiServer and a new instance of the EiSystemBO class. It then defines an enumeration, **e**, and calls **getSystemEnumeration** to retrieve a list of system information. System fields are delimited by commas, as shown below.

```
SBYN,SeeBeyond,A,404 Huntington Dr,null,Monrovia,CA,91016,
null,Los Angeles,null,Western,2001-08-01 15:14:20.0,10,
false,AA#-###-####,false,null,false
CBC,Cape Burr Center,A,1259 Shoreline Drive,null,Cape
Burr,CT,09876,null,Cape Burr,USA,South Central,2002-05-01
17:23:05.0,11,false,###-###-###-##,false,null,true
```

```
EiServer eiServer = new EiServer("EiServer.properties");
EiSystemBO system = eiServer.getEiBOFactory().getEiSystemBOInstance();
Enumeration e = systemBO.getSystemEnumeration();
while (e.hasMoreElements())
    System.out.println(e.nextElement());
    ...
```

EnumCodeType Class

Description

The **EnumCodeType** class defines the code types and represents the enumerations used by **com.stc.eIndex.active.person.CodeLookup**.

Properties

The **EnumCodeType** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
   |
   └─com.stc.eIndex.active.person.EnumCodeType
```

Constructor

None.

Fields

The fields included in the **EnumCodeType** class are described on the following pages:

- **RELIGION** on page 4-358
- **STATE** on page 4-358
- **SUFFIX**on page 4-358
- **SYSTEM** on page 4-359
- **TITLE** on page 4-359
- **VETERAN_STATUS** on page 4-359
- **VIP** on page 4-359

Note: *For a sample of how these fields can be used in your Java programs, see the sample for the* ***getDisplayValue*** *method in the CodeLookup class (beginning on page 4-102).*

Methods

None.

Inherited Methods

The **EnumCodeType** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

ADDRESS_TYPE

Description

The **ADDRESS_TYPE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent address types.

Syntax

```
public static final EnumCodeType ADDRESS_TYPE
```

CITIZENSHIP

Description

The **CITIZENSHIP** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent citizenships.

Syntax

```
public static final EnumCodeType CITIZENSHIP
```

COUNTRY

Description

The **COUNTRY** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent countries.

Syntax

```
public static final EnumCodeType COUNTRY
```

DEPT

Description

The **DEPT** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent departments.

Syntax

```
public static final EnumCodeType DEPT
```

DISTRICT_OF_RESIDENCE

Description

The **DISTRICT_OF_RESIDENCE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent DORs.

Syntax

```
public static final EnumCodeType DISTRICT_OF_RESIDENCE
```

DRIVER_LICENSE_ISSUER

Description

The **DRIVER_LICENSE_ISSUER** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent states or organizations that issue driver licenses.

Syntax

```
public static final EnumCodeType DRIVER_LICENSE_ISSUER
```

ETHNICITY

Description

The **ETHNICITY** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent ethnicities.

Syntax

```
public static final EnumCodeType ETHNICITY
```

EVENT

Description

The **EVENT** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent events.

Syntax

```
public static final EnumCodeType EVENT
```

EVENT_NOTIFICATION

Description

The **EVENT_NOTIFICATION** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent events of notification.

Syntax

```
public static final EnumCodeType EVENT_NOTIFICATION
```

GENDER

Description

The **GENDER** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent genders.

Syntax

```
public static final EnumCodeType GENDER
```

LANGUAGE

Description

The **LANGUAGE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent languages.

Syntax

```
public static final EnumCodeType LANGUAGE
```

MARITAL_STATUS

Description

The **MARITAL_STATUS** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent marital statuses.

Syntax

```
public static final EnumCodeType MARITAL_STATUS
```

NATIONALITY

Description

The **NATIONALITY** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent nationalities.

Syntax

```
public static final EnumCodeType NATIONALITY
```

PERSON_CATEGORY

Description

The **PERSON_CATEGORY** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent person categories.

Syntax

```
public static final EnumCodeType PERSON_CATEGORY
```

PERSON_STATUS

Description

The **PERSON_STATUS** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent member statuses.

Syntax

```
public static final EnumCodeType PERSON_STATUS
```

PHONE_TYPE

Description

The **PHONE_TYPE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent telephone number types.

Syntax

```
public static final EnumCodeType PHONE_TYPE
```

RACE

Description

The **RACE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent races.

Syntax

```
public static final EnumCodeType RACE
```

REGION

Description

The **REGION** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent regions.

Syntax

```
public static final EnumCodeType REGION
```

RELIGION

Description

The **RELIGION** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent religions.

Syntax

```
public static final EnumCodeType RELIGION
```

STATE

Description

The **STATE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent states.

Syntax

```
public static final EnumCodeType STATE
```

SUFFIX

Description

The **SUFFIX** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent suffixes.

Syntax

```
public static final EnumCodeType SUFFIX
```

SYSTEM

Description

The **SYSTEM** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent source applications.

Syntax

```
public static final EnumCodeType SYSTEM
```

TITLE

Description

The **TITLE** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent titles.

Syntax

```
public static final EnumCodeType TITLE
```

VETERAN_STATUS

Description

The **VETERAN_STATUS** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent veteran statuses.

Syntax

```
public static final EnumCodeType VETERAN_STATUS
```

VIP

Description

The **VIP** field specifies that the CodeLookup class should look up the data elements in the *stc_common_detail* table that represent VIP flags.

Syntax

```
public static final EnumCodeType VIP
```

EnumCountryOption

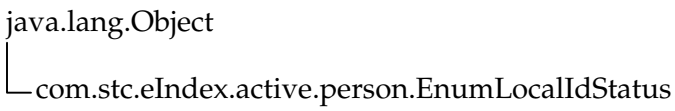
Description

The **EnumCountryOption** class represents an enumeration of country-specific option definitions used by the **CountryOptionLookup** class to retrieve information about country-specific attributes. For an example of how to use the fields in this class, see the example code for **getCountryCode** beginning on page 4-137.

Properties

The **EnumCountryOption** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Fields

The fields included in the **EnumCountryOption** class are described in detail on the following pages:

- **ADDRESS_SEARCH** on page 4-363
- **COLUMN_FORMAT** on page 4-363
- **GROUP_LABEL** on page 4-363
- **SUMMARY_TAB** on page 4-363
- **TAB_LABEL** on page 4-364

Methods

None.

Inherited Methods

The **EnumCountryOption** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

ADDRESS_SEARCH

Description

The **ADDRESS_SEARCH** field specifies that the CountryOptionLookup class should look up country-specific attributes for the *address search* control type.

Syntax

```
public static final EnumCountryOption ADDRESS_SEARCH
```

COLUMN_FORMAT

Description

The **COLUMN_FORMAT** field specifies that the CountryOptionLookup class should look up country-specific attributes for the *column format* control type.

Syntax

```
public static final EnumCountryOption COLUMN_FORMAT
```

GROUP_LABEL

Description

The **GROUP_LABEL** field specifies that the CountryOptionLookup class should look up country-specific attributes for the *group label* control type.

Syntax

```
public static final EnumCountryOption GROUP_LABEL
```

SUMMARY_TAB

Description

The **SUMMARY_TAB** field specifies that the CountryOptionLookup class should look up country-specific attributes for the *summary tab* control type.

Syntax

```
public static final EnumCountryOption SUMMARY_TAB
```

TAB_LABEL

Description

The **TAB_LABEL** field specifies that the CountryOptionLookup class should look up country-specific attributes for the *tab label* control type.

Syntax

```
public static final EnumCountryOption TAB_LABEL
```

EnumLocalIdStatus Class

Description

The **EnumLocalIdStatus** class represents the enumerations used by the **getStatus** method in the LocalId class to retrieve the status of local ID records.

Properties

The **EnumLocalIdStatus** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
   |
   └─com.stc.eIndex.active.person.EnumLocalIdStatus
```

Constructor

None.

Fields

The fields included in the **EnumLocalIdStatus** class are described in detail on the following pages:

- **ACTIVE** on page 4-369
- **DEACTIVATED** on page 4-369
- **MERGED** on page 4-370

Methods

The method included in the **EnumLocalIdStatus** class is described in detail on the following page:

- **getCode** on page 4-367
- **getEnumeration** on page 4-367
- **toString** on page 4-368

Inherited Methods

The **EnumLocalIdStatus** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCode

Description

The **getCode** method retrieves the database code for the status enumeration of a local ID (as defined in e*Index Administrator's Control Key Maintenance function).

Syntax

```
public java.lang.String getCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a coded value	The code for the local ID's status was retrieved successfully.
Null	No coded value was defined for the given status. This indicates an invalid status or an error in the database code table data.

Throws

None.

Example

To see how **getCode** can be used, see the example for **getLocalId** in the LocalId class on page 4-380. This sample retrieves a person's local ID records, and lists the individual elements in each record. **getCode** is used in conjunction with **LocalId.getStatus** to retrieve the code represented by the local ID status enumeration.

getEnumeration

Description

The **getEnumeration** method retrieves the enumeration of the specified status code.

Syntax

```
public static EnumLocalIdStatus getEnumeration(java.lang.String
code)
```

Parameters

Parameter	Description
code	The status code for a local ID.

Return Value

The **getEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration	The enumeration value of the local ID status code was retrieved successfully.

Throws

The **getEnumeration** method throws the following exception:

- **EiException Class**

Example

The following example searches for a UID given a system, local ID, and local ID status. It first retrieves an instance of EiServer and PersonBO, and then calls **getEnumeration** to retrieve the enumeration of the specified status. the example then looks up the UID based on the local ID and system you specified, and the enumeration retrieved for the status.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
...
```

toString

Description

The **toString** method returns a string representation of the local ID status enumeration.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing a status	The local ID status string was retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in the **java.lang.Object** class (see your Java documentation for more information about this method)

ACTIVE

Description

The **ACTIVE** field is used to retrieve an enumeration for the local ID status **A** (active).

Syntax

```
public static final EnumLocalIdStatus ACTIVE
```

DEACTIVATED

Description

The **DEACTIVATED** field is used to retrieve an enumeration for the local ID status **D** (deactivated).

Syntax

```
public static final EnumLocalIdStatus DEACTIVATED
```

MERGED

Description

The **MERGED** field is used to retrieve an enumeration for the local ID status **M** (merged).

Syntax

```
public static final EnumLocalIdStatus MERGED
```

EnumPersonStatus Class

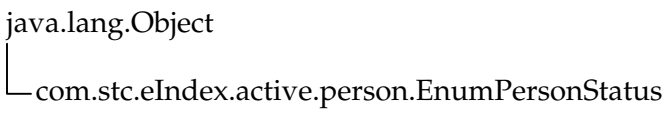
Description

The **EnumPersonStatus** class represents the enumerations used by the **getStatus** method in the DemographicsRO class to retrieve the status of person records.

Properties

The **EnumPersonStatus** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Fields

The fields included in the **EnumPersonStatus** class are described in detail on the following pages:

- **ACTIVE** on page 4-374
- **DEACTIVATED** on page 4-374
- **MERGED** on page 4-375

Methods

The method included in the **EnumPersonStatus** class is described in detail on the following page:

- **getEnumeration** on page 4-373
- **toString** on page 4-373

Inherited Methods

The **EnumPersonStatus** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getEnumeration

Description

The **getEnumeration** method retrieves the enumeration value of the specified status code.

Syntax

```
public static EnumPersonStatus getEnumeration(java.lang.String code)
```

Parameters

Parameter	Description
Code	The status code for the person record.

Return Value

The **getEnumeration** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration	The enumeration value of the person record status code was retrieved successfully.

Throws

The **getEnumeration** method throws the following exception:

- **EiException Class**

toString

Description

The **toString** method returns a string representation of the status.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing a status	The status description was retrieved successfully for the specified status code.

Throws

None.

Additional Information

Overrides:

- **toString** in the **java.lang.Object** class (see your Java documentation for more information about this method)

ACTIVE

Description

The **ACTIVE** field is used to retrieve the enumeration value for the person record status code **A** (active).

Syntax

```
public static final EnumPersonStatus ACTIVE
```

DEACTIVATED

Description

The **DEACTIVATED** field is used to retrieve the enumeration value for the person record status code **D** (deactivated).

Syntax

```
public static final EnumPersonStatus DEACTIVATED
```

MERGED

Description

The **MERGED** field is used to retrieve the enumeration value for the person record status code **M** (merged).

Syntax

```
public static final EnumPersonStatus MERGED
```

LocalId Class

Description

The **LocalId** class represents a local ID record associated with a person record.

Properties

The **LocalId** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**

java.lang.Object
└com.stc.eIndex.active.core.DataObject
└com.stc.eIndex.active.person.PersonDependent
└com.stc.eIndex.active.person.LocalId

Constructor

The **LocalId** class has one constructor, which is described in detail of the following page:

- **LocalId** on page 4-378

Methods

The methods included in the **LocalId** class are described in detail on the following pages:

- **equals** on page4-379
- **getLocalId** on page 4-379
- **getStatus** on page 4-381
- **getSystemCode** on page 4-381
- **getUid** on page 4-382

Inherited Methods

The **LocalId** class inherits these methods from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **LocalId** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

LocalId

Description

The **LocalId** constructor method creates a new Local ID object.

Syntax

```
public LocalID(java.lang.String systemCode, java.lang.String localId)
```

Parameters

Parameter	Description
systemCode	The processing code of the system associated with the local ID.
localId	The new local ID that is associated with the specified system.

Return Value

The **LocalId** method returns the following value:

This value is returned ...	if this occurs ...
LocalId object	The new Local ID object was created successfully.

Throws

The **LocalId** method throws the following exception:

- **EiException Class**

Example

The following example calls the **LocalId** constructor method to create a new LocalId object based on the system and local ID specified. It then calls **hasLocalId** to see if the specified Person object is already associated with the local ID and system pair. If the Person object is not associated with the new local ID and system, **addLocalId** is called to add the new local ID and system pair to the Person object.

```
...
     LocalId lid = new LocalId(system_code,local_id);
     if (!personBO.hasLocalId(person,lid))
         personBO.addLocalId(person, lid);
...
```

equals

Description

The **equals** method checks if two local ID and system pairs are identical.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The Local ID object to compare against other Local ID objects.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The two local ID and system pairs are identical.
False	The two local ID and system pairs are not identical.

Throws

None.

Additional Information

Overrides:

- **equals** in the **java.lang.Object** class (see your Java documentation for more information about this method)

getLocalId

Description

The **getLocalId** method retrieves the local identifier of the local ID and system record. You can use this method to display the value of the **localId** field.

Syntax

```
public java.lang.String getLocalId()
```

Parameters

Parameter	Description
None	

Return Value

The **getLocalId** method returns the following value:

This value is returned …	if this occurs …
String containing a local identifier	The local identifier of the local ID/system record was retrieved successfully.

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for a given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getLocalIdEnumeration** to obtain a list of all system and local ID pairs associated with the person record.

The sample retrieves the individual fields from each local ID record by calling the '**get**' methods in the LocalId class.

```
...
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration lid = person.getLocalIdEnumeration();
 while (lid.hasMoreElements()) {
    LocalId id = (LocalId)lid.nextElement();
    String idDate = sdf.format(id.getCreateDate());
    System.out.println(" UID: " + id.getUid());
    System.out.println(" System: " + id.getSystemCode());
    System.out.println(" ID: " + id.getLocalId());
    System.out.println(" Status: " + id.getStatus().getCode());
...
```

getStatus

Description

The **getStatus** retrieves a handle to the enumeration that represents the status of the local ID record. You can use this method in conjunction with **getCode** the EnumLocalIdStatus class to display the value of the **status** field.

Syntax

```
public EnumLocalIdStatus getStatus()
```

Parameters

Parameter	Description
None	

Return Value

The **getStatus** method returns the following value:

This value is returned …	if this occurs …
EnumLocalIdStatus object representing a status code	The status of the local ID record was retrieved successfully.

Throws

None.

Example

To see an example of how **getStatus** can be used, see the example for **getLocalId** on page 4-380. Note that in this example, **getStatus** is used in conjunction with **getCode** from the EnumLocalIdStatus class. **getStatus** retrieves a handle to the enumeration, and **getCode** retrieves the actual status code.

getSystemCode

Description

The **getSystemCode** method retrieves the processing code for the system associated with a local identifier. You can use this method to display the system code in a local ID record.

Syntax

```
public java.lang.String getSystemCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getSystemCode** method returns the following value:

This value is returned …	if this occurs …
String containing a system code	The system associated with the local ID record was retrieved successfully.

Throws

None.

Example

To see an example of how **getSystemCode** can be used, see the example for **getLocalId** on page 4-380.

getUid

Description

The **getUid** method retrieves the UID associated with a local ID and system record. You can use this method to display the UID associated with a local ID record.

Syntax

```
public Uid getUid()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid** method returns the following value:

This value is returned …	if this occurs …
Uid object	The UID of the local ID record was retrieved successfully.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-482.

Example

To see an example of how **getUid** can be used, see the example for **getLocalId** on page 4-380.

LocalIdBO Class

Description

The **LocalIdBO** class is the local ID business object class and represents the status of a given local ID and system pair for a specific record.

Properties

The **LocalIdBO** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└──com.stc.eIndex.active.person.LocalIdBO

Constructor

None.

Methods

The methods included in the **LocalIdBO** class are described in detail on the following pages:

- **getLocalIdEnumeration** on page 4-385
- **isActive** on page 4-386
- **isDeactivated** on page 4-387
- **isMerged** on page 4-388
- **lookupLocalId** on page 4-389

Inherited Methods

The **LocalIdBO** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getLocalIdEnumeration

Description

The **getLocalIdEnumeration** method retrieves an enumeration of the local ID records associated with the specified UID. You can also specify the local ID status for the lookup.

Syntax

```
public java.util.Enumeration getLocalIdEnumeration(Uid uid)
```

or

```
public java.util.Enumeration getLocalIdEnumeration(Uid uid,
EnumLocalIdStatus localIdStatus)
```

Parameters

Parameter	Description
uid	The UID associated with the person record whose local ID information you want to retrieve.
localIdStatus	The status of the local IDs you want to retrieve. This parameter is optional. If this parameter is omitted, **getLocalIdEnumeration** retrieves IDs of all statuses.

Return Value

The **getLocalIdEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of Local ID objects	The local ID information was retrieved successfully.

Throws

The **getLocalIdEnumeration** method throws the following exceptions:

- EiException Class
- **java.sql.SQLException**

Example

The example below instantiates EiServer and LocalIdBO, and then defines the variable **uid**. It then calls **getLocalIdEnumeration** in the LocalIdBO class to obtain a list of all Local ID records associated with the given UID. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 LocalIdBO localIdBO = eiServer.getEiBOFactory().getLocalIdBOInstance();
 Uid uid = new Uid("1000000108");

 Enumeration lid = localIdBO.getLocalIdEnumeration(uid);
 while (lid.hasMoreElements()) {
     LocalId id = (LocalId)lid.nextElement();
...
```

isActive

Description

The **isActive** method determines whether a given local ID and system pair is active.

Syntax

```
public boolean isActive(LocalId localId)
```

Parameters

Parameter	Description
localId	The Local ID object containing the local ID and system pair to check.

Return Value

The **isActive** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The local ID and system pair has a status of active.
False	The local ID and system pair does not have a status of active.

Throws

The **isActive** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **LocalId Class**, see page 4-376.

Example

The example on the following page calls **loadPerson** to populate the Person object named **person**. It then retrieves an enumeration of local ID records associated with that person. The example scrolls through the resulting LocalId objects to determine whether each local ID is active, merged, or deactivated. In this example, the results are displayed on the screen. You can use the results of the calls to **isActive**, **isMerged**, and **isDeactivated** to determine how to process a local ID record. For example, if a local ID is deactivated, it cannot be modified; or if a person record has only one active local ID remaining, that local ID cannot be deactivated.

```
... /* Getting class instances, getting new
       Person object person, and defining
       Uid object uid.  */
 LocalIdBO localId = eiServer.getEiBOFactory().getLocalIdBOInstance();
 personBO.loadPerson(uid, person);

 Enumeration e = person.getLocalIdEnumeration();
 while (e.hasMoreElements()) {
     LocalId id = (LocalId)e.nextElement();
     String lid = id.getLocalId();
     boolean a = localIdBO.isActive(id);
     boolean m = localIdBO.isMerged(id);
     boolean d = localIdBO.isDeactivated(id);
     if (a){
         System.out.println("Local ID " + lid + " is Active.");
     } else if (m){
         System.out.println("Local ID " + lid + " is Merged.");
     } else if (d) {
         System.out.println("Local ID " + lid + " is Deactivated.");
     }
 }
...
```

isDeactivated

Description

The **isDeactivated** method determines whether a given local ID and system pair has been deactivated.

Syntax

```
public boolean isDeactivated(LocalId localId)
```

Parameters

Parameter	Description
localId	The Local ID object containing the local ID and system pair to check.

Return Value

The **isDeactivated** method returns one of the following values:

This value is returned …	if this occurs …
True	The local ID and system pair has a status of deactivated.
False	The local ID and system pair does not have a status of deactivated.

Throws

The **isDeactivated** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **LocalId Class**, see page 4-376.

Example

To see an example of how **isDeactivated** can be used, see the example for **isActive** beginning on page 4-387.

isMerged

Description

The **isMerged** method determines whether a given local ID and system pair has been merged.

Syntax

```
public boolean isMerged(LocalId localId)
```

Parameters

Parameter	Description
localId	The Local ID object containing the local ID and system pair to check.

Return Value

The **isMerged** method returns one of the following values:

This value is returned …	if this occurs …
True	The local ID and system pair has a status of merged.
False	The local ID and system pair does not have a status of merged.

Throws

The **isMerged** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **LocalId Class**, see page 4-319.

Example

To see an example of how **isMerged** can be used, see the example for **isActive** beginning on page 4-387.

lookupLocalId

Description

The **lookupLocalId** method retrieves an enumeration of local ID records in a specific system that are associated with the same person record as the given system and local ID. This function looks up the UID for the source system and local ID, and retrieves local IDs associated with that UID that were assigned by the target system. You can optionally specify a status for the local IDs returned by the lookup process. If you don't specify a status, only IDs with a merged or deactivated status are returned.

Syntax

```
public java.util.Enumeration lookupLocalId(java.lang.String
sourceSystem, java.lang.String lid, java.lang.String
targetSystem, EnumLocalIdStatus status)
```

or

```
public java.util.Enumeration lookupLocalId(java.lang.String
sourceSystem, java.lang.String lid, java.lang.String
targetSystem, EnumLocalIdStatus status)
```

Parameters

Parameter	Description
sourceSystem	The system code of the system for which the local ID is known.
lid	The local ID associated with the specified source system.
targetSystem	The system code of the system for which you want to look up local ID records.
status	The status code of the local ID records you want to retrieve. This parameter is optional. If no status is specified, the API looks for IDs with a status of merged or deactivated.

Return Value

The **lookupLocalId** method returns the following value:

This value is returned …	if this occurs …
Enumeration	The local IDs were retrieved successfully.

Throws

The **lookupLocalId** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **EnumLocalIdStatus Class**, see page 4-365.

Example

The following example instantiates EiServer and LocalIdBO, and then defines the variable **uid**. It then calls **lookupLocalId** in the LocalIdBO class to obtain a list of all local ID records that are associated with the UID of the given local ID and system pair, that were assigned by the target system, and that are of the specified status. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");

 LocalIdBO localIdBO = eiServer.getEiBOFactory().getLocalIdBOInstance();
 Uid uid = new Uid("1000000108");

 EnumLocalIdStatus status = EnumLocalIdStatus.getEnumeration(status);
 Enumeration lid = localIdBO.lookupLocalId(sourceSystem, lid,
   targetSystem, status);
 while (lid.hasMoreElements())
     LocalId id = (LocalId)lid.nextElement();
     ...
```

LocalIdGenerator Class

Description

The **LocalIdGenerator** class provides a method of generating the next local ID for the specified system. The methods in this class are only available to Oracle implementations.

Properties

The **LocalIdGenerator** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
    |
    +---com.stc.eIndex.active.person.LocalIdGenerator
```

Constructor

None.

Methods

The methods included in the **LocalIdGenerator** class are described in detail on the following pages:

- **getInstance** on page 4-393
- **getNextLocalId** on page 4-393

Inherited Methods

The **LocalIdGenerator** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getInstance

Description

The **getInstance** method retrieves the single instance of the LocalIdGenerator class for the given EiServer object.

Syntax

```
public static LocalIdGenerator getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the LocalIdGenerator instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of LocalIdGenerator	An instance of the LocalIdGenerator class was retrieved successfully for the specified EiServer object.

Throws

None.

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

To see a sample of how **getInstance** in the LocalIdGenerator class can be used, see the example for **getNextLocalId** on page 4-394.

getNextLocalId

Description

The **getNextLocalId** method retrieves the next available local identifier for the given system.

Syntax

```
public java.lang.String getNextLocalId(java.lang.String
systemCode)
```

Parameters

Parameter	Description
systemCode	The processing code for the system for which the local ID is generated.

Return Value

The **getNextLocalId** method returns the following value:

This value is returned …	if this occurs …
String containing a local identifier	The next local identifier available for the specified system was retrieved successfully.

Throws

The **getNextLocalId** method throws the following exceptions. When the local ID number reaches the number specified in the max_no column in *ui_local_id_generator*, the error message **Next local ID exceeds its defined upper bound.** is returned and an exception is thrown.

- **EiException Class**
- **java.sql.SQLException**

Example

The following example retrieves an instance of EiServer and an instance of LocalIdGenerator for EiServer. It then calls **getNextLocalId** and passes the specified system code as the parameter. If the specified system code is valid, **getNextLocalId** returns the next local ID. If the specified system is invalid, **getNextLocalId** returns null. If no system is passed in, the output for this example is **Require Parameter Missing**.

```
EiServer eiServer = new EiServer("EiServer.properties");
LocalIdGenerator lidGen = LocalIdGenerator.getInstance(eiServer);

String LID = lidGen.getNextLocalId(system_code);
System.out.println(" Next Local ID (" + system_code + "): " + LID);
```

Person Class

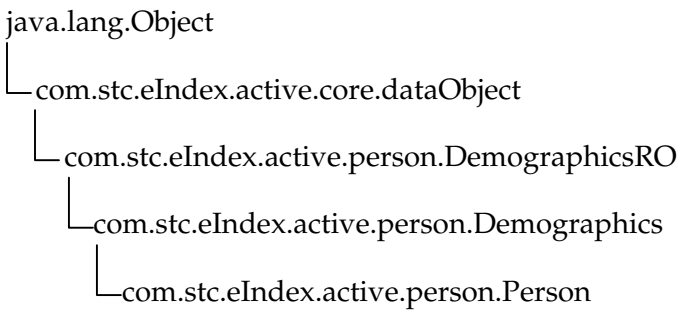
Description

The **Person** class represents a person record and its associated data, such as demographic, alias, local ID, address, telephone, and miscellaneous information. A person record can have several associated local ID, alias, telephone, address, and non-unique ID records.

Properties

The **Person** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.Demographics**



Constructor

The **Person** class has one constructor, which is described on the following page:

- **Person** on page 4-397

Methods

The methods included in the **Person** class are described in detail on the following pages:

- **clearAll** on page 4-398
- **getAddressEnumeration** on page 4-399
- **getAliasEnumeration** on page 4-400
- **getAuxIdEnumeration** on page 4-401
- **getDemographics** on page 4-402
- **getLocalIdEnumeration** on page 4-402
- **getPhoneEnumeration** on page 4-404
- **getUid** on page 4-405
- **setDemographics** on page 4-405

Inherited Methods

The **Person** class inherits all the methods from the classes **Demographics** (except for **clearAll**) and **DemographicsRO Class** (for more information, see the class descriptions on page 4-89 and 4-242).

The **Person** class inherits these methods from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **Person** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Person

Description

The **Person** constructor method creates a new Person object. You can then populate the Person object through different methods, including calls to the Demographics '**set**' methods, **person.setDemographics**, **personBO.addAlias**, **personBO.addLocalId**, and so on.

Syntax

```
public Person()
```

Parameters

Parameter	Description
None	

Return Value

The **Person** method returns one of the following values:

This value is returned …	if this occurs …
Person object	The new Person object was created successfully.

Throws

None.

Example

The following example gets a new instance of EiServer and PersonBO, and then creates a new Person object. The example uses the '**set**' methods (inherited from the Demographics class) to set the values in the new Person object.

```
...
  EiServer eiServer = new EiServer("EiServer.properties");
  PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

  Person person = new Person();
    person.setFirstName(first_name);
    person.setLastname(last_name);
    ... /* Populating the Person object using setter methods */
```

clearAll

Description

The **clearAll** method clears all demographic fields as well as any dependent objects for a Person object, allowing you to re-use the object.

Syntax

```
public void clearAll()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

None.

Additional Information

Overrides:

- **clearAll** in the **Demographics** class (described on page 4-148)

Example

The following example gets a new instance of EiServer and PersonBO, and then creates a new Person object **person**. The example uses the '**set**' methods (inherited from the Demographics class) to set the values in the new Person object. After processing the Person object, the example calls **clearAll** to clear the properties of the object.

```
...
  EiServer eiServer = new EiServer("EiServer.properties");
  PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

  Person person = new Person();
    person.setFirstName(first_name);
    person.setLastname(last_name);
    ... /* Populating the Person object using setter methods */

  ...
  person.clearAll();
...
```

getAddressEnumeration

Description

The **getAddressEnumeration** method retrieves an enumeration of the addresses associated with the specified Person object.

Syntax

```
public java.util.Enumeration getAddressEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getAddressEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of Address objects	The address information was retrieved successfully.

Note: *The* ***getAddressEnumeration*** *method never returns null. If there are no addresses found,* ***getAddressEnumeration*** *returns an empty enumeration.*

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getAddressEnumeration** to obtain a list of all addresses associated with the person record. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
Person person = new Person();

EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);
personBO.loadPerson(uid, person);
Enumeration a = person.getAddressEnumeration();
while (a.hasMoreElements()) {
    Address ad = (Address)a.nextElement();
    ...
```

getAliasEnumeration

Description

The **getAliasEnumeration** method retrieves an enumeration of the aliases associated with the specified Person object.

Syntax

```
public java.util.Enumeration getAliasEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getAliasEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of Alias objects	The alias information was retrieved successfully.

Note: *The* ***getAliasEnumeration*** *method never returns null. If there are no aliases for a person object,* ***getAliasEnumeration*** *returns an empty enumeration.*

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getAliasEnumeration** to obtain a list of all alias names associated with the person record. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
Person person = new Person();

EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);
personBO.loadPerson(uid, person);
Enumeration a = person.getAliasEnumeration();
while (a.hasMoreElements()) {
    Alias an = (Alias)a.nextElement();
    ...
```

getAuxIdEnumeration

Description

The **getAuxIdEnumeration** method retrieves an enumeration of the non-unique IDs (also known as *auxiliary IDs*) associated with a Person object.

Syntax

```
public java.util.Enumeration getAuxIdEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getAuxIdEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of AuxId objects	The non-unique ID information was retrieved successfully.

Note: *The* ***getAuxIdEnumeration*** *method never returns null. If no non-unique IDs are found,* ***getAuxIdEnumeration*** *returns an empty enumeration.*

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getAuxIdEnumeration** to obtain a list of all non-unique IDs associated with the person record. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
Person person = new Person();

EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);
personBO.loadPerson(uid, person);
Enumeration x = person.getAuxIdEnumeration();
while (x.hasMoreElements()) {
    AuxId aux = (AuxId)x.nextElement();
    ...
```

getDemographics

Description

The **getDemographics** method retrieves a DemographicsRO object for a Person object without having to instantiate a new Person object. Used with the **setDemographics** method, the **getDemographics** method can be used to modify the demographic properties of the person record. The DemographicsRO object returned by **getDemographics** can be modified.

Syntax

```
public Demographics getDemographics()
```

Parameters

Parameter	Description
None	

Return Value

The **getDemographics** method returns the following value:

This value is returned …	if this occurs …
Demographics object	The object containing demographic information for a person record was retrieved successfully.

Throws

The **getDemographics** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Demographics Class**, see page 4-144.

Example

To see an example of how **getDemographics** can be used, see the sample code for **setDemographics** beginning on page 4-406.

getLocalIdEnumeration

Description

The **getLocalIdEnumeration** method retrieves an enumeration of the local ID and system pairs associated with the specified Person object.

Syntax

```
public java.util.Enumeration getLocalIdEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getLocalIdEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of LocalId objects	The local ID and system pairs were retrieved successfully.

Note: *The* ***getLocalIdEnumeration*** *method never returns null.*

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getLocalIdEnumeration** to obtain a list of all local ID and system pairs associated with the person record. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
Person person = new Person();

EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);

personBO.loadPerson(uid, person);
Enumeration l = person.getLocalIdEnumeration();
while (l.hasMoreElements()) {
    LocalId lid = (LocalId)lid.nextElement();
    ...
```

getPhoneEnumeration

Description

The **getPhoneEnumeration** method retrieves an enumeration of the telephone numbers associated with the specified Person object.

Syntax

```
public java.util.Enumeration getPhoneEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getPhoneEnumeration** method returns the following value:

This value is returned …	if this occurs …
Enumeration of Phone objects	The telephone information was retrieved successfully.

Note: *The* ***getPhoneEnumeration*** *method never returns null. If there are no telephone numbers found,* ***getPhoneEnumeration*** *returns an empty enumeration.*

Throws

None.

Example

The example below creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by *system_code*, *local_id*, and *status*). It then calls **loadPerson** to retrieve all of the information associated with that UID, and calls **getPhoneEnumeration** to obtain a list of all telephone numbers associated with the person record. It scrolls through the records using **hasMoreElements** and **nextElement**.

```
...
Person person = new Person();

EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
Uid uid = personBO.lookupUid(system_code,local_id,enum);
personBO.loadPerson(uid, person);
Enumeration p = person.getPhoneEnumeration();
while (p.hasMoreElements()) {
    Phone ph = (Phone)p.nextElement();
    ...
```

getUid

Description

The **getUid** retrieves the UID for the specified Person object.

Syntax

```
public Uid getUid()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid** method returns one of the following values:

This value is returned …	if this occurs …
UID object	The UID for the specified Person object was retrieved successfully.

Throws

None.

setDemographics

Description

The **setDemographics** method sets the demographics contained in a Person object. You can used this method to set the fields retrieved by **getDemographics** for the Person object. This method performs a field-by-field copy of all demographic fields.

Syntax

```
public void setDemographics(Demographics demographics)
```

Parameters

Parameter	Description
demographics	A Demographics object containing the information to copy into the Person object.

Return Value

None.

Throws

The **setDemographics** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the Demographics Class, see page 4-144.Example

The following example obtains instances of EiServer and PersonBO, and then looks up a person object by local ID, system, and status. It calls **loadPerson** to load the person information, and calls **getDemographics** to retrieve the demographics associated with the Person object. It then calls the '**set**' methods in the Demographics class to update property information, and calls **setDemographics** to update the new information in the Person object. **updatePerson** is called to update the information in the e*Index database.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_code,local_id,enum);
 personBO.loadPerson(uid, person);

 Demographics demo1 = person.getDemographics();
     demo1.setLastName(last_name);
     demo1.setFirstName(first_name);
     demo1.setMiddleName(middle_name);
     demo1.setGender(gender_code);
     ... /*  Setting remaining demographic properties */

 person.setDemographics(demo1);
 Transaction trans = new Transaction(userId, system, source,
    department, terminalId);
 boolean updatePerformed = personBO.updatePerson(person, trans);
...
```

PersonBO Class

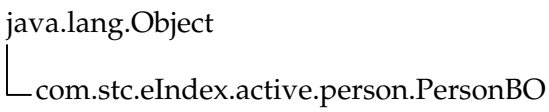
Description

The **PersonBO** class is the business object class for the **Person** class. This class contains the methods that you use to search for and process data in the e*Index database.

Properties

The **PersonBO** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None.

Methods

The methods included in the **PersonBO** class are described in detail on the following pages:

Inherited Methods

The **PersonBO** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **Notify**
- **notifyAll**
- **toString**
- **wait**

addAddress

Description

The **addAddress** method adds a new address to a Person object. Before adding an address to an existing record, you should call **hasAddress** to verify that the person record does not already have an address of the same type.

Syntax

```
public void addAddress(Person person, Address address)
```

Parameters

Parameter	Description
person	The Person object to which the address information will be added.
address	The Address object containing the information to add to the Person object.

Return Value

None.

Throws

The **addAddress** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Address Class**, see page 4-33.

Example

The following example loads a Person object from the database and then creates a new Address object (**add**) of the type specified by the variable *address_type*. It then sets the address information using the '**set**' methods in the Address class. The example checks if the Person object is already associated with an address type equal to the address type of **add**. If there is no address of the same type, the new Address object is added to the Person object. If there is an address of the same type, the example retrieves an enumeration of address records associated with the Person object and checks each address in the enumeration to find the address record whose type matches that of the new Address object **add**. When that address record is found, **equals** is called to see if the two Address objects are identical. If they are not identical, the existing record is updated with the information from **add**. If they are identical, no processing is performed.

```
...
 Person person = new Person();
 personBO.loadPerson(new Uid(uid_number), person);

 Address add = new Address(address_type);
   add.setAddress1(street_address1);
   add.setAddress2(street_address2);
   add.setAddress3(street_address3);
   add.setAddress4(street_address4);
   add.setCity(city);
   add.setStateOrProvince(state);
   add.setCounty(county);
   add.setPostalCode(zip_code);
   add.setPostalCodeExt(zip_code_extension);
   add.setCountry(country);
 if (personBO.hasAddress(person,add)) {
   Enumeration enum = person.getAddressEnumeration();
   if (enum.hasMoreElements()) {
     while (enum.hasMoreElements()) {
       Address exist_add = (Address)enum.nextElement();
       if (exist_add.getAddressType().equals(add.getAddressType())) {
          boolean a_equals = (exist_add.equals(add));
          if (!a_equals) {
             exist_add.setAddress1(add.getAddress1());
             exist_add.setAddress2(add.getAddress2());
             exist_add.setAddress3(add.getAddress3());
             exist_add.setAddress4(add.getAddress4());
             exist_add.setCity(add.getCity());
             exist_add.setStateOrProvince(add.getStateOrProvince());
             exist_add.setCounty(add.getCounty());
             exist_add.setPostalCode(add.getPostalCode());
             exist_add.setPostalCodeExt(add.getPostalCodeExt());
             exist_add.setCountry(add.getCountry());
          }
       }
     }
   }
 } else {
   personBO.addAddress(person,add);
 }
 ...
```

addAlias

Description

The **addAlias** method adds an alias name to a person record. Before adding an alias to an existing record, you should call **hasAlias** to verify that the alias name is not already associated with that record.

Syntax

```
public void addAlias(Person person, Alias alias)
```

Parameters

Parameter	Description
person	The Person object to which the alias information will be added.
alias	The Alias object containing the information to add to the Person object.

Return Value

None.

Throws

The **addAlias** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Alias Class**, see page 4-58.

Example

The following example creates a new Alias object with the specified last, first, and middle names (specified by the variables *last_name*, *first_name*, and *middle_name*). It then calls **hasAlias** to see if the specified Person object is already associated with the new alias information. If the Person object is not associated with the alias, **addAlias** is called to add the new alias information to the Person object.

```
...
  Alias a = new Alias(last_name,first_name,middle_name);
  if (!personBO.hasAlias(person,a)) {
      personBO.addAlias(person, a);
...
```

addAudit

Description

The **addAudit** method adds an audit log entry to a person record.

Syntax

```
public void addAudit(Person person, Audit audit)
```

Parameters

Parameter	Description
person	The Person object to which the audit information will be added.
audit	The Audit object containing the information to add to the Person object.

Return Value

None.

Throws

The **addAudit** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Audit Class**, see page 4-70.

Example

The following example creates and populates a new Person object. Before calling **addPerson**, a new Audit object is created and populated with information from the user application, and then **addAudit** is called to add the Audit record to the Person object.

```
...
 Person person = new Person();
 /* Populating the person object using setter methods  */

 Audit audit = new Audit();
     audit.setUserId(application_user_id);
     audit.setFunct(function);
     audit.setDetail(function_detail);
     audit.setTimestamp(eiServer.getDbTimestamp());
 personBO.addAudit(person, audit);
...
```

addAuxId

Description

The **addAuxId** method adds a non-unique (auxiliary) ID to a person record. Before adding a non-unique ID to an existing record, call **hasAuxId** to verify that the ID is not already associated with that record.

Syntax

```
public void addAuxId(Person person, AuxId auxId)
```

Parameters

Parameter	Description
person	The Person object to which the non-unique ID will be added.
auxId	The AuxId object containing the information to add to the Person object.

Return Value

None.

Throws

The **addAuxId** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **AuxId Class**, see page 4-81.

Example

The following example creates and populates a new Person object. It then creates a new AuxId object with the specified non-unique ID type and number. The example calls **hasAuxId** to check if the Person object already contains a matching AuxId object. If a matching AuxId object is not found, **addAuxId** is called to add the new AuxId object to the Person object.

```
...
 Person person = new Person();
 /* Populating the person object using setter methods  */

 AuxId a = new AuxId(id_type, id_number);
 if (!personBO.hasAuxId(person,a)) {
     personBO.addAuxId(person, a);
...
```

addLocalId

Description

The **addLocalId** method adds a local ID and system record to a person record. Before adding a local ID to an existing record, you should call **hasLocalId** to verify that the local ID is not already associated with that record.

Syntax

```
public void addLocalId(Person person, LocalId localId)
```

Parameters

Parameter	Description
person	The Person object to which the local ID information will be added.
localId	The Local ID object containing the information to add to the Person object.

Return Value

None.

Throws

The **addLocalId** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **LocalId Class**, see page 4-376.

Example

The following example creates a new LocalId object with system code and local ID specified by the variables *system* and *local_id*. It then calls **hasLocalId** to see if the specified Person object is already associated with the local ID and system pair. If the Person object is not associated with the new local ID and system, **addLocalId** is called to add the new local ID and system pair to the Person object.

```
...
     LocalId lid = new LocalId(system,local_id);
     if (!personBO.hasLocalId(person,lid))
         personBO.addLocalId(person, lid);
...
```

addPerson

Description

The **addPerson** method adds a new person record to the e*Index database. You can specify whether the method should perform a potential duplicate check when adding the record.

Syntax

```
public Uid addPerson(Person person, Transaction trans,
boolean checkForDuplicates)
```

Parameters

Parameter	Description
person	The Person object to add as a new record to the database.
trans	The transaction object for the current transaction.
checkForDuplicates	A Boolean flag that specifies whether the application should check for potential duplicate records when adding the new person record. When the flag is set to **#t**, potential duplicates are checked. When the flag is set to **#f**, potential duplicates are not checked.

Return Value

The **addPerson** method returns one of the following values:

This value is returned …	if this occurs …
UID object	The person record was successfully added to the e*Index database.

Throws

The **addPerson** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Transaction Class**, see page 4-482.

Example

The following example gets instances of EiServer and PersonBO, and then creates a new Person object named **person**. The example calls the '**set**' methods (inherited from Demographics) to populate the demographic fields in **person**, and then creates a new LocalId object **lid** to populate local ID

information in **person**. The example then creates a new Transaction object **trans**, and calls **addPerson** to insert a new record into the database.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

 Person person = new Person();
     person.setLastName(last_name);
     person.setFirstName(first_name);
     person.setMiddleName(middle_name);
     ... /* Setting person properties */

    LocalId lid = new LocalId(system, id);
    personBO.addLocalId(person, lid);

    Transaction trans = new Transaction(userId, system, function,
     source, department, terminalId);
    boolean dupchkFlag = false;

    Uid uid = personBO.addPerson(person, trans, dupchkFlag);
...
```

addPhone

Description

The **addPhone** method adds a telephone number to a person record. Before adding a telephone number to an existing record, you should call **hasPhone** to verify that the number is not already associated with that record.

Syntax

```
public void addPhone(Person person, Phone phone)
```

Parameters

Parameter	Description
person	The Person object to which the non-unique ID will be added.
phone	The Phone object containing the information to add to the Person object.

Return Value

None.

Throws

The **addPhone** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Phone Class**, see page 4-452.

Example

The following example loads a Person object from the database and then creates a new Phone object (**ph**) of the type specified by the variable *phone_type*. It then sets the phone information using the '**set**' methods in the Phone class. The example checks if the Person object is already associated with a phone type equal to the phone type of **add**. If there is no matching phone type, the new Phone object is added to the Person object. If there is a matching phone type, the example retrieves an enumeration of phone records associated with the Person object and checks each record in the enumeration to find one whose type matches that of the new Phone object. When that record is found, **equals** is called to see if the two Phone objects are identical. If they are not identical, the existing record is updated with the information from **ph**. If they are identical, no processing is performed.

```
...
 Person person = new Person();
 personBO.loadPerson(new Uid(uid_number), person);
 Phone ph = new Phone(phone_type);
   ph.setPhoneNumber(phone_number);
   ph.setPhoneExtension(phone_extension);

 if (personBO.hasPhone(person,ph)) {
   Enumeration enum = person.getPhoneEnumeration();
   if (enum.hasMoreElements()) {
      while (enum.hasMoreElements()) {
         Phone exist_ph = (Phone)enum.nextElement();
         if (exist_ph.getPhoneType().equals(ph.getPhoneType())) {
            boolean equals = (exist_ph.equals(ph));
            if (!equals) {
               exist_ph.setPhoneNumber(ph.getPhoneNumber());
               exist_ph.setPhoneExtension(ph.getPhoneExtension()); }
         }
      }
   }
 } else {
     personBO.addPhone(person,ph); }
...
```

addPredefinedComment

Description

The **addPredefinedComment** method adds predefined comment to a person record.

Syntax

```
public void addPredefinedComment(Person person,
java.lang.String predefinedMsgCode)
```

Parameters

Parameter	Description
person	The Person object to which the predefined comment will be added.
predefinedMsgCode	The code that indicates the predefined message to add to the Person object.

Return Value

None.

Throws

The **addPredefinedComment** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395.

Example

The following example creates and populates a new Person object, and then defines the variable *msgCode* as the predefined message code **REV**. It then calls **addPredefinedComment** to add the predefined message to the Person object.

```
...
 Person person = new Person();
.../* Populating person record */

 String msgCode = "REV";
 personBO.addPredefinedComment(person, predefinedMsgCode);
...
```

addUserDefinedComment

Description

The **addUserDefinedComment** method adds free-text comment to a person record.

Syntax

```
public void addPredefinedComment(Person person,
java.lang.String comment)
```

Parameters

Parameter	Description
person	The Person object to which the user-defined comment will be added.
comment	The text of the comment to add to the Person object.

Return Value

None.

Throws

The **addUserDefinedComment** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395.

Example

The following example creates and populates a new Person object, and then defines the variable *comment* as user-entered text. It then calls **addUserDefinedComment** to add the message to the Person object.

```
...
 Person person = new Person();
.../* Populating person record */

 String comment = "Record is flagged for deactivation.";
 personBO.addUserDefinedComment(person, comment);
...
```

getActiveUid

Description

The **getActiveUid** method follows a merge path for a specific record, and retrieves the active person record for the given UID. If the person record associated with the given UID has a merged status, this method allows you to retrieve the active record.

Syntax

```
public Uid getActiveUid(Uid uid)
```

Parameters

Parameter	Description
uid	The UID of the specified record.

Return Value

The **getActiveUid** method returns one of the following values:

This value is returned …	if this occurs …
UID object	The active record for the given UID was retrieved successfully. If the given UID was active, then the given UID is returned.
Null	The UID passed to **getActiveUid** was active, deactivated, or non-existant.

Throws

The **getActiveUid** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

The following example retrieves the active UID associated with the specified UID. First it retrieves an instance of the EiServer and PersonBO classes, and then creates a new Uid object. The sample then calls **getActiveUid** to lookup the active UID associated with the UID that was entered as a parameter. Once you retrieve the active UID, you can use it to perform actions against the active record, such as updating demographic information. In this example, **getDemographics** is called after **getActiveUid** to retrieve the demographic information for the active UID.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Uid uid = new Uid("1000005917");
 DemographicsRO person;

     Uid uid = personBO.getActiveUid(uid);
     person = personBO.getDemographics(uid);
...
```

getDemographics

Description

The **getDemographics** method retrieves the demographic information for the person record associated with the given UID, returning a DemographicsRO object. The method performs this function without needing to instantiate a new Person object.

Syntax

```
public DemographicsRO getDemographics(Uid uid)
```

Parameters

Parameter	Description
uid	The UID of the specified record.

Return Value

The **getDemographics** method returns one of the following values:

This value is returned …	if this occurs …
DemographicsRO object	The demographic information for the person record was retrieved successfully.

Throws

The **getDemographics** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

To see an example of how **getDemographics** in the **PersonBO** class can be used, see the example provided for **getActiveUid** on page 4-420.

hasAddress

Description

The **hasAddress** method determines if the given address already exists for the specified Person object.

Syntax

```
public boolean hasAddress(Person person, Address address)
```

Parameters

Parameter	Description
person	The Person object whose addresses you want to check.
address	The Address object being compared against the addresses that already exist for the specified Person object.

Return Value

The **hasAddress** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The address already exists in the Person object.
False	The address does not exist in the Person object.

Throws

The **hasAddress** method throws the following exceptions:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Address Class**, see page 4-33.

Example

To see an example of how **hasAddress** in the **PersonBO** class can be used, see the example provided for **addAddress** on page 4-409.

hasAlias

Description

The **hasAlias** method determines if the given alias name already exists for the specified Person object.

Syntax

```
public boolean hasAlias(Person person, Alias alias)
```

Parameters

Parameter	Description
person	The Person object whose aliases you want to check.
alias	The alias name being compared against the alias names that already exist for the specified Person object.

Return Value

The **hasAlias** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The alias name already exists in the Person object.
False	The alias name does not exist in the Person object.

Throws

The **hasAlias** method throws the following exceptions:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Alias Class**, see page 4-58.

Example

To see an example of how **hasAlias** in the **PersonBO** class can be used, see the example provided for **addAlias** on page 4-411.

hasAuxId

Description

The **hasAuxId** method determines if the given non-unique ID already exists for the specified Person object.

Syntax

```
public boolean hasAuxId(Person person, AuxId auxId)
```

Parameters

Parameter	Description
person	The Person object whose aliases you want to check.
auxId	The non-unique ID being compared against the non-unique IDs that already exist for the specified Person object.

Return Value

The **hasAuxId** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The non-unique ID already exists in the Person object.
False	The non-unique ID does not exist in the Person object.

Throws

The **hasAuxId** method throws the following exceptions:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **AuxId Class**, see page 4-81.

Example

To see an example of how **hasAuxId** in the **PersonBO** class can be used, see the example provided for **addAuxId** on page 4-413.

hasLocalId

Description

The **hasLocalId** method determines if the given local ID and system pair already exists for the specified Person object.

Syntax

```
public boolean hasLocalId(Person person, LocalId localId)
```

Parameters

Parameter	Description
person	The Person object whose local ID information you want to check.
localId	The local ID record being compared against the local ID records associated with the specified Person object.

Return Value

The **hasLocalId** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The given local ID and system pair already exists in the specified Person object.
False	The given local ID and system pair does not exist in the specified Person object.

Throws

The **hasLocalId** method throws the following exceptions:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **LocalId Class**, see page 4-376.

Example

To see an example of how **hasLocalId** in the **PersonBO** class can be used, see the example provided for **addLocalId** on page 4-414.

hasPhone

Description

The **hasPhone** method determines if the given telephone number already exists for the specified Person object.

Syntax

```
public boolean hasPhone(Person person, Phone phone)
```

Parameters

Parameter	Description
person	The Person object whose telephone numbers you want to check.

Parameter	Description
phone	The Phone object being compared against the telephone numbers that already exist for the specified Person object.

Return Value

The **hasPhone** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The telephone number already exists in the Person object.
False	The telephone number does not exist in the Person object.

Throws

The **hasPhone** method throws the following exceptions:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Phone Class**, see page 4-452.

Example

To see an example of how **hasPhone** in the **PersonBO** class can be used, see the example provided for **addPhone** on page 4-417.

isActivePerson

Description

The **isActivePerson** method checks to see if the specified person record is active.

Syntax

```
public boolean isActivePerson(Person person)
```

Parameters

Parameter	Description
person	The Person object whose status you want to check.

Return Value

The **isActivePerson** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The specified person record has a status of **A** (active).
False	The specified person record was found and it has a status of **M** (merged) or **D** (deactivated).

Throws

The **isActivePerson** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395.

Example

In the following example, **psr** represents a PersonSearchResult object returned from a demographic search, and **dr** represents the DemographicsResultRO objects included in **psr**. After the search is performed, the example scrolls through and loads each returned record, and calls **isActivePerson**, **isMergedPerson**, and **isDeactivatedPerson** to determine the status of each record. For active records, the UID and demographic information is displayed. For merged and deactivated records, the UID is displayed along with a message stating the record's status.

```
EiServer eiServer = new EiServer("EiServer.properties");
PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
ControlKey contrlKey = ControlKey.getInstance(eiServer);

.../* defining search parameters in object searchParams */

 PersonSearchResult psr = null;
 DemographicsResultRO dr = null;
 psr = personBO.searchAlpha(sp);

 System.out.println("Matching Records: " + psr.size());
 if (psr==null) {
   System.out.println("No matching records.");
 } else {
   while(psr.next()) {
     dr = psr.getResultRow();
     Uid uid = dr.getUid();
     Person person = new Person();
     personBO.loadPerson(uid, person);
     boolean active = personBO.isActivePerson(person);
     boolean merge = personBO.isMergedPerson(person);
     boolean deact = personBO.isDeactivatedPerson(person);
     if (active){
       System.out.println("UID: " + uid + "  Demographics: " + dr);
     } else if (merge){
       System.out.println("Record with Uid: " +uid + " is Merged.");
     } else if (deact){
       System.out.println("Record with Uid: " +uid + " is Deactivated.");
     }
   }
 }
  ...
```

isDeactivatedPerson

Description

The **isDeactivatedPerson** method checks to see if the specified person record has been deactivated in the e*Index database.

Syntax

```
public boolean isDeactivatedPerson(Person person)
```

Parameters

Parameter	Description
person	The Person object whose status you want to check.

Return Value

The **isDeactivatedPerson** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The specified person record has a status of **D** (deactivated).
False	The specified person record was found and it has a status of **M** (merged) or **A** (active).

Throws

The **isDeactivatedPerson** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395.

Example

For an example of how **isDeactivatedPerson** can be used, see the example for **isActivePerson** on page 4-427.

isMergedPerson

Description

The **isMergedPerson** method checks to see if the specified person record has been merged into another record.

Syntax

```
public boolean isMergedPerson(Person person)
```

Parameters

Parameter	Description
person	The Person object whose status you want to check.

Return Value

The **isMergedPerson** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The specified person record has a status of **M** (merged).
False	The specified person record was found and it has a status of **A** (active) or **D** (deactivated).

Throws

The **isMergedPerson** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Person Class**, see page 4-395.

Example

For an example of how **isMergedPerson** can be used, see the example for **isActivePerson** on page 4-427.

loadPerson

Description

The **loadPerson** method populates a Person object with the information associated with a person record in the database. Once you load a Person object, you can display and perform updates to the person information.

Syntax

```
public void loadPerson(Uid uid, Person person)
```

Parameters

Parameter	Description
uid	The UID object containing the UID of the person record to be loaded.
person	The Person object to load the person information into.

Return Value

None.

Throws

The **loadPerson** method throws the following exception:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Uid Class**, see page 4-493. For more information about the **Person Class**, see page 4-395.

Example

The following example gets a new instance of EiServer and Person BO, and then creates a new Person object named **person**. The example calls **lookupUid** to find a UID based on a user-specified system, local ID, and status combination. It then calls **loadPerson** to populate **person** with demographic, alias, and local ID information.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();

 EnumLocalIdStatus status = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_name,local_id,status);
 personBO.loadPerson(uid, person);
...
```

lookupUid

Description

The **lookupUid** method searches for a record's UID based on the criteria specified. You can specify one of two different combinations of criteria:

- Social security number (SSN)
- Local ID, system, and status

Syntax

```
public java.util.Enumeration lookupUid(Ssn ssn)
```

or

```
public Uid lookupUid(java.lang.String systemCode,
java.lang.String localId, EnumLocalIdStatus status)
```

Parameters

Parameter	Description
ssn	The SSN of the person whose UID you want to retrieve.
systemCode	The processing code of the system associated with the specified local ID. This parameter is case-sensitive.
localId	The local ID of the person whose UID you want to retrieve.
status	The status of the specified local ID record.

Return Value

The **lookupUid** method returns one of the following values:

This value is returned …	if this occurs …
UID object or Enumeration	A UID record associated with the specified SSN or local ID and system was found.
Null	No UID record was found matching the specified criteria.

Throws

The **lookupUid** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Ssn Class**, see page 4-478. For more information about the **Uid Class**, see page 4-493.

Example

The following example calls **LookupUid** to search for a UID based on the local ID, system, and local ID status specified. The example retrieves an instance of EiServer and PersonBO, and then defines status using a call to **getEnumeration** to obtain the enumeration value of the specified status. It calls **lookupUid** to find the person record, and displays the resulting UID.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 EnumLocalIdStatus status = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system_name,local_id,status);
 System.out.println("Uid found via Local Id search: " + uid);
...
```

The following is similar to the previous example except it searches for the UID record using the person's social security number.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Enumeration e = personBO.lookupUid(new Ssn(social_security_number));
 System.out.println("UIDs found via SSN search:");
 while (e.hasMoreElements()) {
     System.out.println(e.nextElement());
...
```

processPerson

Description

The **processPerson** method adds a new person record to the e*Index database or updates an existing record. The logic used to perform the add or update transaction is based on standard e*Index processing, which is described in "About Inbound Event Processing Logic" in Chapter 2 of the *e*Index Global Identifier Technical Reference*.

Syntax

```
public Uid processPerson(Person person, Transaction trans)
```

Parameters

Parameter	Description
person	The Person object containing the new information to be added to or updated in the e*Index database.
trans	The Transaction object containing information for the current transaction.

Return Value

The **processPerson** method returns the following value:

This value is returned …	if this occurs …
UID object	The person record was successfully added to the e*Index database and assigned a unique identifier.

Throws

The **processPerson** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Transaction Class**, see page 4-482.

Example

The following example gets instances of EiServer and PersonBO, and then creates a new Person object named **person**. The example calls the '**set**' methods (inherited from Demographics) to populate the demographic fields in **person**, and then creates a new LocalId object **lid** to populate local ID information in **person**. The example then creates a new Transaction object **trans**, and calls **processPerson** to either insert a new record into the database or update an existing record in the database (following standard e*Index processing logic).

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

 Person person = new Person();
     person.setLastName(last_name);
     person.setFirstName(first_name);
     person.setMiddleName(middle_name);
     ... /* Setting person properties */

 LocalId lid = new LocalId(system,id);
    personBO.addLocalId(person, lid);

 Transaction trans = new Transaction(userId, system, source,
      department, terminalId);

 Uid uid = personBO.processPerson(person, trans);
...
```

searchAlpha

Description

The **searchAlpha** method performs an alphanumeric search against the e*Index database based on the specified search criteria. This method performs an exact search on last names when the length of the specified last name is equal to or less than the value of the LNEXTSRCH control key. Otherwise, it performs a partial search on last names. If the SEEMERGED and SEEDEACTIVATED control keys are set to **N**, this search only returns active records. For more information about the LNEXTSRCH control key, see chapter 5 of the *e*Index Administrator User's Guide*. For more information about standard alphanumeric searches, see chapter 3 of the *e*Index Global Identifier User's Guide.*

Syntax

```
public PersonSearchResult searchAlpha(SearchParameters
searchParms)
```

Parameters

Parameter	Description
searchParms	The search parameters to use for the current alphanumeric search.

Return Value

The **searchAlpha** method returns one of the following values:

This value is returned …	if this occurs …
List of records	Records matching the specified criteria were found and retrieved successfully.
Null	No matching records were found.

Throws

The **searchAlpha** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **SearchParameters Class**, see page 4-470.

Example

The following example performs an alphanumeric search based on the search criteria you specify. First the example gets an instance of EiServer and defines a SimpleDateFormat variable **sdf**. It then defines the variable **sp** as a

new SearchParameters object. The example checks the arguments entered and sets the values using the '**set**' calls in SearchParameters (in this case, the symbol % indicates a null value). Once the object **sp** is populated, the example gets a new PersonBO instance, and initializes the variables **psr** and **dr** (of the type PersonSearchResult and DemographicsResultRO, respectively). **searchAlpha** is called using the parameters that were populated into the SearchParameters object **sp**. The search criteria are displayed using the '**get**' calls in SearchParameters and then the results of the search are displayed, using the DemographicsResult object **dr** to get the UID of each record. To see other ways to implement a search, see **PersonSearch.java** in the sample code provided with the Java API. This file is located in **/<home_dir>/sample/com/stc/eIndex/active/sample**.

```
EiServer eiServer = new EiServer("EiServer.properties");
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");

SearchParameters sp = new SearchParameters();
      sp.setLastName(last_name);
      sp.setFirstName(first_name);
      sp.setMiddleName(middle_name);
      sp.setGender(gender);
      sp.setDob(new java.sql.Date(sdf.parse(date_of_birth).getTime()));
      sp.setSsn(new Ssn(social_security_number));
      sp.setMotherMaidenName(mother_maiden_name);
      sp.setMaidenName(maiden_name);

   PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
   PersonSearchResult psr = null;
   DemographicsResultRO dr = null;
   psr = personBO.searchAlpha(sp);

   System.out.println("SEARCH CRITERIA USED:");
   System.out.println("  Name: " + sp.getLastName() + ", "
        + sp.getFirstName() + " " + sp.getMiddleName() + "  - Gender: "
        + sp.getGender() + "  - DOB: " + sp.getDob() + "  Mother's MN: "
        + sp.getMotherMaidenName() + "  -  Mother's Name: "
        + sp.getMaidenName());
      if (sp.getSsn() != null)
          System.out.println("  SSN: " + sp.getSsn().toString());
      else
          System.out.println("  SSN:  null");

   System.out.println("MATCHING RECORDS: ");
    while(psr.hasNext())
     dr = psr.next();
      System.out.println("UID:" + dr.getUid() + " Demographics: " + dr);
...
```

searchGeneral

Description

The **searchGeneral** method performs a general, alphanumeric search against the e*Index database based on the specified search criteria. For a general search, you can use any combination of data as long as one data element is stored in an indexed database column. The search criteria for a general search can contain wildcards. To indicate a wildcard, enter a percent sign (%). If the SEEMERGED and SEEDEACTIVATED control keys are set to **N**, this search only returns active records. For more information about general searches, see chapter 3 of the *e*Index Global Identifier User's Guide*.

Syntax

```
public PersonSearchResult searchGeneral(SearchParameters
searchParms)
```

Parameters

Parameter	Description
searchParms	The search parameters to use for the current general search.

Return Value

The **searchGeneral** method returns one of the following values:

This value is returned …	if this occurs …
List of records	Records matching the specified criteria were found and retrieved successfully.
Null	No matching records were found.

Throws

The **searchGeneral** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **SearchParameters Class**, see page 4-470.

Example

The following example performs a general search based on the search criteria you specify. First the example gets an instance of EiServer and defines a SimpleDateFormat variable **sdf**. It then defines the variable **sp** as a new SearchParameters object. The example checks the arguments entered and sets the values using the '**set**' calls in SearchParameters (in this case, the

symbol % indicates a null value). Once the object **sp** is populated, the example gets a new PersonBO instance, and initializes the variables **psr** and **dr** (of the type PersonSearchResult and DemographicsResultRO, respectively). **searchGeneral** is called using the parameters that were populated into the SearchParameters object **sp**. The search criteria are displayed using the '**get**' calls in SearchParameters and then the results of the search are displayed, using the DemographicsResult object **dr** to get the UID of each record.

```
EiServer eiServer = new EiServer("EiServer.properties");
SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");

SearchParameters sp = new SearchParameters();
   sp.setLastName("WARREN");
   sp.setFirstName("ELIZABETH");
   sp.setMaritalStatus("M");
   sp.setSpouseName("DAVID");
   ... /* Setting remaining SearchParameters properties.*/

PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
PersonSearchResult psr = psr = personBO.searchGeneral(sp);
System.out.println("MATCHING RECORDS: ");
if (psr==null) {
    System.out.println("No Results");
} else {
    while (psr.next()) {
        System.out.println(psr.getResultRow());
    }
}
...
```

searchPhonetic

Description

The **searchPhonetic** method performs a phonetic search against the e*Index database based on the specified search criteria. Remember that the possible parameters you can use in a phonetic search are limited to specific combinations. If the SEEMERGED and SEEDEACTIVATED control keys are set to **N**, this search only returns active records. For more information about the possible combinations, see chapter 3 of the *e*Index Global Identifier User's Guide*.

Syntax

```
public PersonSearchResult searchPhonetic(SearchParameters
searchParms)
```

Parameters

Parameter	Description
searchParms	The search parameters to use for the current phonetic search.

Return Value

The **searchPhonetic** method returns one of the following values:

This value is returned …	if this occurs …
List of records	Records matching the specified criteria were found and retrieved successfully.
Null	No matching records were found.

Throws

The **searchPhonetic** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **SearchParameters Class**, see page 4-470.

Example

searchPhonetic is called in the same manner as **searchAlpha**. For more information, see the example for **searchAlpha** on page 4-435. An example is also provided in **PersonSearch.java** in the e*Index sample Java files. This file is located in **/<home_dir>/sample/com/stc/eIndex/active/sample**.

uidLookupByAuxId

Description

The **uidLookupByAuxId** method searches the database for UIDs associated with the non-unique ID and type specified. For more information about non-unique ID searches, see chapter 3 of the *e*Index Global Identifier User's Guide*.

Syntax

```
public java.util.Enumeration uidLookupByAuxId(java.lang.String
auxIdType, java.lang.String auxId)
```

Parameters

Parameter	Description
auxIdType	The processing code of the non-unique ID type associated with the specified ID.
auxId	The non-unique identification code of the person(s) whose UIDs you want to retrieve.

Return Value

The **uidLookupByAuxId** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration	An enumeration of UIDs associated with the specified non-unique ID and type was found.
Null	No UID record was found matching the specified criteria.

Throws

The **uidLookupByAuxId** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

The following example calls **uidLookupByAuxId** to search for UIDs based on the ID type and ID specified (*id_type* and *id_number*). The example retrieves an instance of EiServer and PersonBO, and calls **uidLookupByAuxId** to retrieve an enumeration of UIDs.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Enumeration enum = personBO.uidLookupByAuxId(id_type, id_number);
    while (enum.hasMoreElements()) {
        Uid uid = (Uid)enum.nextElement();
    }...
```

updatePerson

Description

The **updatePerson** method updates an existing person record in the database. If the DUPCHK control key is enabled and a key search field is modified, **updatePerson** performs a duplicate check upon updating the person record.

Syntax

```
public boolean updatePerson(Person person, Transaction trans)
```

Parameters

Parameter	Description
person	The Person object to be updated.
trans	The Transaction object containing information about the current transaction.

Return Value

The **updatePerson** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	An existing database record was updated.
False	No updates were required to the existing record (the record remained unchanged).

Throws

The **updatePerson** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

For more information about the **Person Class**, see page 4-395. For more information about the **Transaction Class**, see page 4-482.

Example

The following example gets instances of EiServer and PersonBO, and then creates a new Person object named **person**. The example then loads the person record to be updated, and calls '**set**' methods to populate the new values for the demographic fields. It creates a new LocalId object **lid** to add new local ID information to the record. The example then creates a new Transaction object **trans**, and calls **updatePerson** to update the loaded Person object in the database.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

 Person person = new Person();
 personBO.loadPerson(new Uid(UID_number), person);

    person.setLastName(last_name);
    person.setFirstName(first_name);
    person.setMiddleName(middle_name);
    ... /* Setting new person properties */

    LocalId lid = new LocalId(system, id);
    personBO.addLocalId(person, lid);

    Transaction trans = new Transaction(userId, system,
     source, department, terminalId);

    boolean update = personBO.updatePerson(person, trans);
    if (update)
        System.out.println("Record was updated.");
    else
        System.out.println("No updates required.");
...
```

PersonSearchResult Class

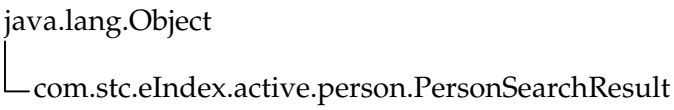
Description

The **PersonSearchResult** class represents a set of DemographicsResultRO objects returned as a result of a demographic search. This object maintains a cursor pointing to the current row of data. Initially, the cursor is positioned before the first row, and the **next** method moves the cursor to the next row. When there are no more rows in the object, **next** returns false so it can be used in a **while** loop to iterate through the result set. You can also reposition the pointer using the **previous**, **first**, **last**, and **setResultRow** methods.

Properties

The **PersonSearchResult** class has the following properties:

- Public final class
- Extends **java.lang.Object**



Constructor

None.

Methods

The methods included in the **PersonSearchResult** class are described in detail on the following pages:

Inherited Methods

The **PersonSearchResult** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

first

Description

The **first** method sets the pointer to the first record in a PersonSearchResult object. When a PersonSearchResult object is populated, the pointer is initially set to the first record. Use the set method to return the cursor to the first record once the pointer has been moved from the initial position.

Syntax

```
public void first()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

None.

Example

The usage of the **first** method is similar to that of the **last** method. To see a sample of how **last** can be used in your Java programs, see the example for **last** on page 4-446.

getResultRow

Description

The **getResultRow** method retrieves the DemographicsResultRO object to which the cursor is currently pointing in the PersonSearchResult object.

Syntax

```
public DemographicsResultRO getResultRow()
```

Parameters

Parameter	Description
None	

Return Value

The **getResultRow** method returns the following value:

This value is returned …	if this occurs …
DemographicsResultRO object	The current DemographicsResultRO object in the PersonSearchResult object was retrieved successfully.

Throws

The **getResultRow** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **DemographicsRO Class**, see page 4-242.

Example

To see an example of how **getResultRow** can be used in your Java programs, see the example for **next** on page 4-448.

last

Description

The **last** method sets the pointer to the last record in a PersonSearchResult object.

Syntax

```
public void last()
```

Parameters

Parameter	Description
None	

Return Value

None.

Throws

None.

Example

The following example creates a new SearchParameters object **sp** and populates the object. It then initializes the variable **psr** (a PersonSearchResult

object), and calls **searchAlpha** to perform a search. The example calls **last** to set the pointer to the last row in **psr** and calls **previous** to scroll through the records in reverse order. It calls **getResultRow** to display each record.

```
...
SearchParameters sp = new SearchParameters();
... /* creating search parameters */

PersonSearchResult psr = null;
psr = personBO.searchAlpha(sp);

if (psr==null) {
    System.out.println("No matching records.");
} else {
   psr.last();
   while (psr.previous()) {
        System.out.println(psr.getResultRow());
...
```

next

Description

The **next** method moves the pointer to the next row in a PersonSearchResult object and indicates whether the row is valid or the end of the PersonSearchResult object has been reached.

Syntax

```
public boolean next()
```

Parameters

Parameter	Description
None	

Return Value

The **next** method returns one of the following Boolean values:

This value is returned …	if this occurs …
true	The pointer was moved to the next record, which was a valid record in the PersonSearchResult object.
false	There were no more rows in the PersonSearchResult object.

Throws

None.

Example

The following example creates a new SearchParameters object **sp** and populates the object. It then initializes the variable **psr** (a PersonSearchResult object), and calls **searchAlpha** to perform a search. The example displays the records returned from the search, using **next** to scroll through the records in the object and **getResultRow** to display each record.

```
...
SearchParameters sp = new SearchParameters();
... /* creating search parameters */
PersonSearchResult psr = null;
psr = personBO.searchAlpha(sp);
if (psr==null) {
    System.out.println("No matching records.");
} else {
    while(psr.next()) {
        System.out.println(psr.getResultRow());
...
```

previous

Description

The **previous** method moves the pointer to the previous row in a PersonSearchResult object and indicates whether the row is valid or the beginning of the PersonSearchResult object has been reached.

Syntax

```
public boolean previous()
```

Parameters

Parameter	Description
None	

Return Value

The **previous** method returns one of the following Boolean values:

This value is returned …	if this occurs …
true	The pointer was moved to the previous record, which was a valid record in the PersonSearchResult object.

This value is returned …	if this occurs …
false	There were no previous rows in the PersonSearchResult object.

Throws

None.

Example

To see a sample of how **previous** can be used in your Java programs, see the example for **last** on page 4-446.

setResultRow

Description

The **setResultRow** method sets the pointer to the specified row number in a PersonSearchResult object. Rows are numbered beginning with one (1).

Syntax

```
public boolean setResultRow(int row)
```

Parameters

Parameter	Description
row	An integer indicating the row number to which the pointer should be set.

Return Value

The **setResultRow** method returns one of the following Boolean values:

This value is returned …	if this occurs …
true	The pointer was moved to the specified row, which was a valid row in the PersonSearchResult object.
false	The row specified was invalid.

Throws

None.

Example

The following example creates a new SearchParameters object **sp** and populates the object. It then initializes the variable **psr** (a PersonSearchResult object), and calls **searchAlpha** to perform a search on the criteria. The example then calls **size** to determine how many records are in the result set, and **setResultRow** to select the specified row number. It then displays the selected record.

```
...
SearchParameters sp = new SearchParameters();
... /* creating search parameters */

PersonSearchResult psr = null;
psr = personBO.searchAlpha(sp);

if (psr==null) {
    System.out.println("No matching records.");
} else {
    int size = psr.size();
        System.out.println(size + " records returned.");
        psr.setResultRow(row_number);
        System.out.println(psr.getResultRow());
...
```

size

Description

The **size** method returns the number of rows contained in a PersonSearchResult object.

Syntax

```
public int size()
```

Parameters

Parameter	Description
None	

Return Value

The **size** method returns the following value:

This value is returned …	if this occurs …
Integer	The number of rows in the PersonSearchResult object was returned successfully.

Throws

None.

Example

To see a sample of how **size** can be used in your Java programs, see the example for **setResultRow** on the previous page.

Phone Class

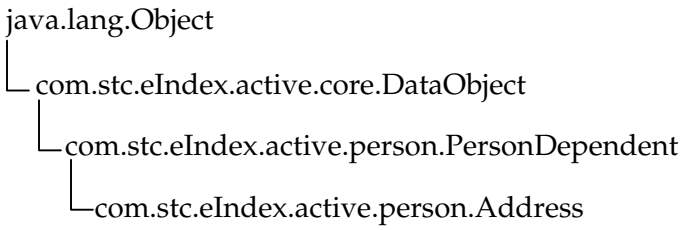
Description

The **Phone** class represents telephone information associated with a person record. To display each field in a Phone object, call the '**get**' methods defined in this class. To populate fields in a Phone object, call the '**set**' methods.

Properties

The **Phone** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.PersonDependent**



Constructor

The **Phone** class has one constructor, which is described on the following page:

- **Phone** on page 4-454

Methods

The methods included in the **Phone** class are described in detail on the following pages:

- **equals** on page 4-455
- **getPhoneExtension** on page 4-456
- **getPhoneNumber** on page 4-457
- **getPhoneType** on page 4-458
- **getUid** on page 4-459
- **setPhoneExtension** on page 4-460
- **setPhoneNumber** on page 4-460
- **toString** on page 4-461

Inherited Methods

The **Phone** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Phone

Description

The **Phone** method is the constructor method for the Phone class. Use this method to create objects containing a person's telephone information.

Syntax

```
public Phone(java.lang.String type)
```

Parameters

Parameter	Description
type	The processing code for the type of telephone number you are adding to the person record, such as **CH** (home), **CO** (office), and so on.

Return Value

The **Phone** constructor method returns the following value:

This value is returned …	if this occurs …
An object containing telephone information	The Phone object was created successfully.

Throws

The **Phone** constructor throws the following exception:

- **EiException Class**

Example

The following example is excerpted from a sample used to create a new Person object and then add demographic, address, telephone, and local ID information to the new object. It retrieves new instances of EiServer and PersonBO, and creates an empty Person object. The example then creates a Demographics object to populate the demographic fields of the Person object, and an Address object to populate the address information. Finally, it creates an empty Phone object, and calls the '**set**' methods to fill the Phone object with telephone information. **addPhone** is called to add the Phone object to the Person object.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();

 Demographics demo = new Demographics();
     demo.setLastName(last_name);
     demo.../* Populating property values */
 ...

 Address address = new Address("H");
     address.setAddress1(address_street1);
     address.../* Populating property values */
 ...

 Phone ph = new Phone("CB");
     ph.setPhoneNumber(telephone_number);
     ph.setPhoneExtension(telephone_extension);
 personBO.addPhone(person, phone);
 ...
```

equals

Description

The **equals** method compares two Phone objects to see if they are identical.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The Phone object to compare against the current Phone object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The two Phone objects are identical.
False	The two Phone objects are not identical.

Throws

None.

Additional Information

Overrides:

- **equals** in class java.lang.Object (see your Java documentation for more information)

Example

To see an example of how the **equals** methods can be used, see the example for the **addPhone** method on page 4-417.

getPhoneExtension

Description

The **getPhoneExtension** method retrieves the value of the **phoneExtension** field in a telephone record. You can use this method to display the extension to a telephone number in a Phone object.

Syntax

```
public java.lang.String getPhoneExtension()
```

Parameters

Parameter	Description
None	

Return Value

The **getPhoneExtension** method returns the following value:

This value is returned …	if this occurs …
String containing the telephone extension	The **phoneExtension** field of the telephone record was retrieved successfully.
Null	The **phoneExtension** field was checked successfully but there was no value to retrieve.

Throws

None.

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the user-defined *system_code*, *local_id*, and *status* variables). It then calls **loadPerson** to retrieve the information associated with that UID, and calls **getPhoneEnumeration** to retrieve a list of all telephone numbers associated

with the person record. The example calls **hasMoreElements** and **nextElement** to scroll through the enumeration, and calls the '**get**' methods in the Phone class to display the fields in each record. Note that the **getPhoneType** function retrieves the coded value of the phone type, so **getDisplayValue** in the CodeLookup class is called to translate the coded value to the display value.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();
 EnumLocalIdStatus enum = EnumLocalIdStatus.getEnumeration(status);
 Uid uid = personBO.lookupUid(system,local_id,enum);
 personBO.loadPerson(uid, person);

 Enumeration p = person.getPhoneEnumeration();
 while (p.hasMoreElements()) {
    Phone ph = (Phone)p.nextElement();
    EnumCodeType codeType = EnumCodeType.PHONE_TYPE;
     String ptype = ph.getPhoneType();
     String display_value = codeLookup.getDisplayValue(codeType, ptype);
     System.out.println(" UID: " + ph.getUid());
     System.out.println(" Phone Type: " + display_value);
     System.out.println(" Phone Number: " + ph.getPhoneNumber());
     System.out.println(" Extension: " + ph.getPhoneExtension());
...
```

getPhoneNumber

Description

The **getPhoneNumber** method retrieves the value of the **phoneNumber** field in a telephone record. You can use this method to display the telephone number in a Phone object.

Syntax

```
public java.lang.String getPhoneNumber()
```

Parameters

Parameter	Description
None	

Return Value

The **getPhoneNumber** method returns the following value:

This value is returned …	if this occurs …
String containing the telephone number	The **phoneNumber** field of the telephone record was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from a Phone object, see the example for the **getPhoneExtension** method on page 4-456.

getPhoneType

Description

The **getPhoneType** method retrieves the value of the **phoneType** field in a telephone record. You can use this method to display the type of telephone number contained in a Phone object.

Syntax

```
public java.lang.String getPhoneType()
```

Parameters

Parameter	Description
None	

Return Value

The **getPhoneType** method returns the following value:

This value is returned …	if this occurs …
String containing the phone type code	The **phoneType** field of the telephone record was retrieved successfully.

Throws

None.

Example

To see an example of how the '**get**' methods can be used to retrieve information from a Phone object, see the example for the **getPhoneExtension** method on page 4-456.

getUid

Description

The **getUid** method retrieves the value of the **uid** field for a telephone record. You can use this method to display the UID associated with a Phone object.

Syntax

```
public Uid getUid()
```

Parameters

Parameter	Description
None	

Return Value

The **getUid** method returns the following value:

This value is returned …	if this occurs …
Uid	The **uid** associated with the telephone record was retrieved successfully.

Throws

None.

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

To see an example of how the '**get**' methods can be used to retrieve information from a Phone object, see the example for the **getPhoneExtension** method on page 4-456.

setPhoneExtension

Description

The **setPhoneExtension** method is the setter for the **phoneExtension** field. Use this method to populate the extension of a person's telephone number in a Phone object.

Syntax

```
public void setPhoneExtension(java.lang.String phoneExtension)
```

Parameters

Parameter	Description
phoneExtension	The value of the **phoneExtension** field for the new Phone object.

Return Value

None.

Throws

The **setPhoneExtension** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Phone object, see the example for the **Phone** method on page 4-454.

setPhoneNumber

Description

The **setPhoneNumber** method is the setter for the **phoneNumber** field. Use this method to populate a person's telephone number in a Phone object.

Syntax

```
public void setPhoneExtension(java.lang.String phoneNumber)
```

Parameters

Parameter	Description
phoneNumber	The value of the **phoneNumber** field for the new Phone object.

Return Value

None.

Throws

The **setPhoneNumber** method throws the following exception:

- **EiException Class**

Example

To see an example of how the '**set**' methods can be used to populate a Phone object, see the example for the **Phone** method on page 4-454.

toString

Description

The **toString** method returns a string representation of the Phone object.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String representing a telephone record	The string representation was returned successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in class **com.stc.eIndex.active.core.DataObject**

Example

The following example creates a new Person object, and then looks up the UID for the given system, local ID, and status (as specified by the user-defined *system_code*, *local_id*, and *status* variables). The example retrieves an enumeration of Phone objects associated with the loaded Person object, and scrolls through each telephone record using the **nextElement** and

hasMoreElements functions. It then calls the **toString** method to display a comma-delimited string representation of each telephone record. Each record is comma-delimited and surrounded by brackets as shown in the following example list.

```
[CB,9895551946,321]
[CH,9895551313,null]
[CC,3235557961,null]
```

```
...
 EiServer eiServer = new EiServer("EiServer.properties");
 PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();
 Person person = new Person();

 Uid uid = personBO.lookupUid(system,local_id,status);
 personBO.loadPerson(uid, person);

 Enumeration p = person.getPhoneEnumeration();
 while (p.hasMoreElements()) {
     Phone ph = (Phone)p.nextElement();
     System.out.println(ph.toString());
...
```

PredefinedMsg Class

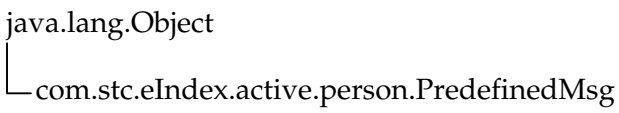
Description

The **PredefinedMsg** class represents the information in a predefined message, and is used by the PredefinedMsgRegistry class to look up message code and description pairs.

Properties

The **PredefinedMsg** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None

Methods

The methods included in the **PredefinedMsg** class are described in detail on the following pages:

- **getCode** on page 4-464
- **getDescription** on page 4-464

Inherited Methods

The **PredefinedMsg** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCode

Description

The **getCode** method retrieves the code for a specific predefined message from the table *ui_canned_msg*. Use this method to display a predefined message code.

Syntax

```
public java.lang.String getCode()
```

Parameters

Parameter	Description
None	

Return Value

The **getCode** method returns one of the following values:

This value is returned …	if this occurs …
String containing a message code	The value of the predefined message code was retrieved successfully.
Null	The predefined message code was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see a sample of how **getDescription** can be used, see the example for **getInstance** in the PredefinedMsgRegistry class beginning on page 4-468.

getDescription

Description

The **getDescription** method retrieves the message text that corresponds with a specific predefined message code from the table *ui_canned_msg*. Use this method to display a predefined message description.

Syntax

```
public java.lang.String getDescription()
```

Parameters

Parameter	Description
None	

Return Value

The **getDescription** method returns one of the following values:

This value is returned …	if this occurs …
String containing a predefined message	The predefined message description was retrieved successfully.
Null	The predefined message description was checked successfully but there was no value to retrieve.

Throws

None.

Example

To see a sample of how **getDescription** can be used, see the example for **getInstance** in the PredefinedMsgRegistry class beginning on page 4-468.

PredefinedMsgRegistry Class

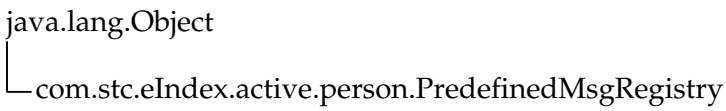
Description

The **PredefinedMsgRegistry** class is a utility class for looking up all predefined messages or for retrieving an enumeration of PredefinedMsgRegistry objects from *ui_canned_msg*. You can then use the methods in the PredefinedMsg class to retrieve the code and text for each message.

Properties

The **PredefinedMsgRegistry** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

None

Methods

The methods included in the **PredefinedMsgRegistry** class are described in detail on the following pages:

- **getInstance** on page 4-467
- **getPredefinedMsgEnumeration** on page 4-468

Inherited Methods

The **PredefinedMsgRegistry** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getInstance

Description

The **getInstance** method retrieves the single instance of the PredefinedMsgRegistry class for the given EiServer object, maintaining a singleton pattern so database results are stored for servicing subsequent lookups.

Syntax

```
public static PredefinedMsgRegistry getInstance(EiServer
eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the PredefinedMsgRegistry instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of the PredefinedMsg Registry class	The PredefinedMsgRegistry instance for the specified EiServer object was retrieved successfully.

Throws

None.

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

The following example retrieves an instance of the EiServer class and an instance of the PredefinedMsgRegistry class. It then retrieves an enumeration of predefined messages and scrolls through each message using the **hasMoreElements** and **nextElement** functions. **getCode** and **getDescription** in the PredefinedMsg Class are called to display the text for each message and their corresponding message codes.

```
...
 EiServer eiServer = new EiServer("EiServer.properties");

 PredefinedMsgRegistry pm;
 pm = PredefinedMsgRegistry.getInstance(eiServer);
 Enumeration e = pm.getPredefinedMsgEnumeration();
 if (e != null) {
    while (e.hasMoreElements()) {
      PredefinedMsg m = (PredefinedMsg)e.nextElement();
      System.out.println("Message Code: " + m.getCode());
      System.out.println("Message:  " + m.getDescription());
    }
 }
...
```

getPredefinedMsgEnumeration

Description

The **getPredefinedMsgEnumeration** method retrieves an enumeration of PredefinedMsg objects. Use this method to display the available predefined message descriptions and codes.

Syntax

```
public java.util.Enumeration getPredefinedMsgEnumeration()
```

Parameters

Parameter	Description
None	

Return Value

The **getPredefinedMsgEnumeration** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration of PredefinedMsg objects	The predefined message codes and descriptions were retrieved successfully.
Null	There were no PredefinedMsg objects to retrieve.

Throws

None.

Example

To see a sample of how **getPredefinedMsgEnumeration** can be used, see the example for **getInstance** beginning on page 4-468.

SearchParameters Class

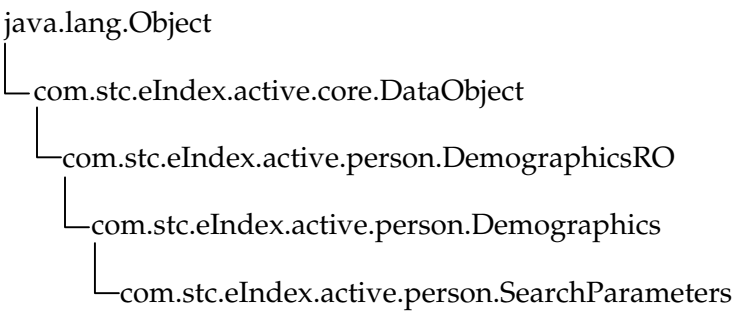
Description

The **SearchParameters** class is an object representing the search parameters that are passed to the PersonBO **searchAlpha, searchGeneral**, and **searchPhonetic** functions.

Properties

The **SearchParameters** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.person.Demographics**



Constructor

The **SearchParameters** class has one constructor, which is described on the following page:

- **SearchParameters** on page 4-472

Methods

The methods included in the **SearchParameters** class are described in detail on the following pages:

- **setUid** on page 4-473

Inherited Methods

The **SearchParameters** class inherits all methods defined in the **DemographicsRO Class** (see page 4-242 for information about these Java methods) and all methods defined in the **Demographics Class** (see page 4-144 for information about these Java methods) .

The **SearchParameters** class inherits this method from **com.stc.eIndex.active.core.DataObject**:

- **toString**

The **SearchParameters** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

SearchParameters

Description

The **SearchParameters** constructor method constructs the search parameters for the **searchAlpha**, **searchGeneral**, and **searchPhonetic** functions in the PersonBO class. The SearchParameters method can take no parameters, one parameter, or eight parameters. If you call SearchParameters with no parameters, you open a new SearchParameters object that you can populate using the **set** methods defined in the **Demographics** class. If you call the method using one parameter, a DemographicsRO object, the DemographicsRO object is used to populate the demographic search fields in the SearchParameters object (this only populates the search fields defined for SearchParameters). Finally, you can call SearchParameters using the eight primary demographic search criteria fields as parameters.

Syntax

```
public SearchParameters()
```

or

```
public SearchParameters(DemographicsRO demo)
```

or

```
public SearchParameters(java.lang.String lastName,
java.lang.String firstName, java.lang.String middleName,
java.lang.String gender, java.sql.Date dob, Ssn ssn,
java.lang.String motherMaidenName, java.lang.String maidenName)
```

Parameters

Parameter	Description
demo	A list of demographic information contained in a DemographicsRO object to use as search criteria.
lastName	The last name in the person record you want to find.
firstName	The first name in the person record you want to find.
middleName	The middle name or initial in the person record you want to find.
gender	The gender of the person whose record you want to find.
dob	The date of birth of the person whose person record you want to find.
ssn	The social security number of the person whose record you want to find.
motherMaidenName	The maiden name of the mother of the person whose record you want to find.

Parameter	Description
maidenName	The maiden name of the person whose record you want to find. This parameter is not used for phonetic searches.

Return Value

The **SearchParameters** method returns the following value:

This value is returned …	if this occurs …
SearchParameters object	The object containing the specified search parameters was created successfully.

Throws

The **SearchParameters** method throws the following exception:

- **EiException Class**

Example

For examples of how you can implement **SearchParameters**, see the example for **searchAlpha** on pages 4-435. An example is also provided in **PersonSearch.java** in the e*Index sample Java files. This file is located in **/<home_dir>/sample/com/stc/eIndex/active/sample**.

setUid

Description

The **setUid** method is the setter method for the **uid** field. Use this method to define the UID for the SearchParameters object when performing a General Search.

Syntax

```
public void setUid(Uid uid)
```

Parameters

Parameter	Description
uid	The new value for the **uid** field.

Return Value

None.

Throws

The **setUid** method throws the following exception:

- **EiException Class**

Additional Information

For more information about the **Uid Class**, see page 4-493.

Example

The following example retrieves an instance of EiServer and PersonBO. It then creates a new SearchParameters object, **sp**, and populates it using '**set**' methods in the SearchParameters and Demographics classes. It then calls **searchGeneral** to perform a general alphanumeric search.

```
...
  EiServer eiServer = new EiServer("EiServer.properties");
  PersonBO personBO = eiServer.getEiBOFactory().getPersonBOInstance();

  SearchParameters sp = new SearchParameters();
     sp.setUid("1001051005");
     sp.setLastName("WARREN");
     sp.setFirstName("ELIZABETH");
     sp.setMaritalStatus("M");
     sp.setSpouseName("DAVID");

  PersonSearchResult psr = psr = personBO.searchGeneral(sp);
...
```

Security Class

Description

The **Security** class provides the security routines for your e*Index active integration implementation.

Properties

The **Security** class has the following properties:

- Public class
- Extends **java.lang.Object**

java.lang.Object
└com.stc.eIndex.active.person.Security

Constructor

None.

Methods

The methods included in the **Security** class is described in detail on the following page:

- **getInstance** on page 4-476
- **isUserValid** on page 4-477

Inherited Methods

The **Security** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getInstance

Description

The **getInstance** method retrieves the single instance of the Security class for the given EiServer object, maintaining a singleton pattern so database results are stored for servicing subsequent lookups.

Syntax

```
public static Security getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the Security instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of the Security class	The Security instance for the specified EiServer object was retrieved successfully.

Throws

None.

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

To see a sample of how **getInstance** in the Security class can be used, see the example for **isUserValid** on page 4-477.

isUserValid

Description

The **isUserValid** method checks the database to verify that the specified user login ID exists in the *stc_user* table, and that the ID is currently active in the table.

Syntax

```
public boolean isUserValid(java.lang.String userID)
```

Parameters

Parameter	Description
userID	The user login ID of the person logging onto e*Index. This parameter is case-sensitive.

Return Value

The **isUserValid** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The user login ID was found in the *stc_user* table and is currently active.
False	The user login ID was not found in the e*Index security tables or is not currently active.

Throws

The **isUserValid** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Additional Information

The **isUserValid** method does not verify the access permissions assigned to the user. It is assumed that each user of the applications you create require access to all functions provided.

Example

The following example retrieves an instance of the EiServer class and an instance of the Security class. It then calls **isUserValid** to find out if the user name provided is defined in the *user_tbl* table in the e*Index database. **isUserValid** returns Boolean true or false. Once the user is validated, you can define how processing should be handled.

```
EiServer eiServer = new EiServer("EiServer.properties");
Security security = Security.getInstance(eiServer);

boolean valid = security.isUserValid(user_login_id);
System.out.println("Testing USER ID: " + user_login_id);
  if (valid)
      ...
```

Ssn Class

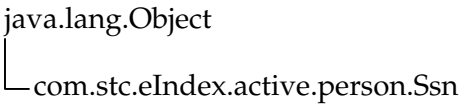
Description

The **Ssn** class is a representation of a person's social security number. Use this class to insert a new social security number into a person record.

Properties

The **Ssn** class has the following properties:

- Public class
- Extends **java.lang.Object**



Constructor

The **Ssn** class has one constructor, which is described on the following page:

- **Ssn** on page 4-479

Field

The field included in the **Ssn** class is described on the following page:

- **NULL** on page 4-481

Methods

The methods included in the **Ssn** class are described in detail on the following pages:

- **equals** on page 4-480
- **toString** on page 4-480

Inherited Methods

The **Ssn** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Ssn

Description

The **Ssn** constructor method creates a new social security number object. Use this method to add a new social security number to a Person object, a SearchParameters object, and so on.

Syntax

```
public Ssn ssn(java.lang.String ssn)
```

Parameters

Parameter	Description
ssn	The value of the social security number for the new object.

Return Value

The **Ssn** method returns one of the following values:

This value is returned …	if this occurs …
Ssn object	The social security number object was created successfully.

Throws

The **Ssn** method throws the following exception:

- **EiException Class**

Example

The following example creates a new SearchParameters object, **sp**, and then populates the search parameter fields using the '**set**' methods in SearchParameters. To set the SSN field, the example creates a new Ssn object by calling **getSsn (new Ssn(*social_security_number*))**.

```
...
 SearchParameters sp = new SearchParameters();
       sp.setLastName(last_name);
       sp.setFirstName(first_name);
       sp.setMiddleName(middle_name);
       sp.setGender(gender);
       sp.setDob(sdf.parse(date_of_birth));
       sp.setSsn(new Ssn(social_security_number));
       sp.setMotherMaidenName(mother_maiden_name);
       sp.setMaidenName(maiden_name);
...
```

equals

Description

The **equals** method compares two Ssn objects to see if they are identical.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The Ssn object to compare against the current Ssn object.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The two Ssn objects are identical.
False	The two Ssn objects are not identical.

Throws

None.

Additional Information

Overrides:

- **equals** in class java.lang.Object (see your Java documentation for more information)

toString

Description

The **toString** method returns a string representation of the Ssn object.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns one of the following values:

This value is returned …	if this occurs …
String containing an SSN	The string representation was returned successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in class java.lang.Object (see your Java documentation for more information)

NULL

Description

The **NULL** field specifies that the field to be set to null is of the type *Ssn*.

Syntax

```
public static final Ssn NULL
```

Transaction Class

Description

The **Transaction** class stores all transaction parameters, such as user ID, system, terminal ID, and so on.

Properties

The **Transaction** class has the following properties:

- Public class
- Extends **com.stc.eIndex.active.core.DataObject**

java.lang.Object
└com.stc.eIndex.active.core.DataObject
└com.stc.eIndex.active.person.Transaction

Constructor

The **Transaction** class has one constructor, which is described on the following page:

- **Transaction** on page 4-484

Methods

The methods included in the **Transaction** class are described in detail on the following pages:

- **equals** on page 4-485
- **getDepartment** on page 4-485
- **getFunction** on page 4-487
- **getSource** on page 4-487
- **getSystem** on page 4-488
- **getTerminalId** on page 4-489
- **getTimestamp** on page 4-490
- **getUserId** on page 4-490
- **toString** on page 4-491

Inherited Methods

The **Transaction** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

Transaction

Description

The **Transaction** constructor method initiates a new transaction. You need to create a new transaction object before processing any person records with calls to **addPerson**, **processPerson**, or **updatePerson**.

Syntax

```
public Transaction(java.lang.String userId, java.lang.String
system, java.lang.String source, java.lang.String department,
java.lang.String terminalId)
```

Parameters

Parameter	Description
userId	The user logon ID of the user performing the transaction. This parameter is required.
system	The system in which the transaction originates. This parameter is required.
source	The system in which the transaction originates. This parameter is optional.
department	The department from which the transaction is performed. This parameter is optional.
terminalId	The identification code of the terminal from which the transaction is performed. This parameter is optional.

Return Value

The **Transaction** method returns the following value:

This value is returned …	if this occurs …
Transaction object	The transaction object was created successfully.

Throws

The **Transaction** method throws the following exception:

- **EiException Class**

Example

The following example creates and populates a new Person object, and the creates a new Transaction object, **trans**, using parameters supplied by the user application. It then calls **addPerson** to complete the transaction, and calls the '**get**' methods in the Transaction class to display the transaction information.

```
...
 Person person = new Person;
  /* Populating person information... */

 Transaction trans = new Transaction(user_id, system, source,
    department, terminal_id);
 boolean checkDuplicate = true;
 Uid uid = personBO.addPerson(person, trans, checkDuplicate);
    System.out.println("USER ID:     " + trans.getUserId());
    System.out.println("SYSTEM:    " + trans.getSystem());
    System.out.println("EVENT:       " + trans.getFunction());
    System.out.println("SOURCE:      " + trans.getSource());
    System.out.println("DEPARTMENT:  " + trans.getDepartment());
    System.out.println("TERMINAL ID: " + trans.getTerminalId());
    System.out.println("CREATEDATE:  " + trans.getTimestamp());
...
```

equals

Description

The **equals** method compares two Transaction objects to see if they are identical.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The object to compare against.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The two Transaction objects are identical.
False	The two Transaction objects are not identical.

Throws

None.

Additional Information

Overrides:

- **equals** in class java.lang.Object (see your Java documentation for more information)

getDepartment

Description

The **getDepartment** method retrieves the value of the **department** field for the transaction. This value represents the department from which the transaction originated. Use this method to display transaction information.

Syntax

```
public java.lang.String getDepartment()
```

Parameters

Parameter	Description
None	

Return Value

The **getDepartment** method returns one of the following values:

This value is returned …	if this occurs …
String containing a department code	The value of the **department** field was retrieved successfully.
Null	No value was associated with the **department** field in the transaction.

Throws

None.

Example

For an example of how the **getDepartment** method can be used, see the example for **Transaction** on page 4-484.

getFunction

Description

The **getFunction** method retrieves the value of the **function** field for the transaction. This value represents the type of event that caused the transaction. Use this method to display transaction information after a record is processed. This method returns null prior to a transaction.

Syntax

```
public java.lang.String getFunction()
```

Parameters

Parameter	Description
None	

Return Value

The **getFunction** method returns one of the following values:

This value is returned …	if this occurs …
String containing an event code	The value of the **function** field was retrieved successfully.
Null	No value was associated with the **function** field in the transaction.

Throws

None.

Example

For an example of how the **getFunction** method can be used, see the example for **Transaction** on page 4-484.

getSource

Description

The **getSource** method retrieves the value of the **source** field for the transaction. This value represents the system from which the transaction originated. Use this method to display transaction information.

Syntax

```
public java.lang.String getSource()
```

Parameters

Parameter	Description
None	

Return Value

The **getSource** method returns one of the following values:

This value is returned …	if this occurs …
String containing a system code	The value of the **source** field was retrieved successfully.
Null	No value was associated with the **source** field in the transaction.

Throws

None.

Example

For an example of how the **getSource** method can be used, see the example for **Transaction** on page 4-484.

getSystem

Description

The **getSystem** method retrieves the value of the **system** field for the transaction. This value represents the system from which the transaction originated. Use this method to display transaction information.

Syntax

```
public java.lang.String getSystem()
```

Parameters

Parameter	Description
None	

Return Value

The **getSystem** method returns the following value:

This value is returned …	if this occurs …
String containing a system code	The value of the **system** field was retrieved successfully.

Throws

None.

Example

For an example of how the **getSystem** method can be used, see the example for **Transaction** on page 4-484.

getTerminalId

Description

The **getTerminalId** method retrieves the value of the **terminalId** field for the transaction. This value represents the terminal at which the transaction was performed. Use this method to display transaction information.

Syntax

```
public java.lang.String getTerminalId()
```

Parameters

Parameter	Description
None	

Return Value

The **getTerminalId** method returns one of the following values:

This value is returned …	if this occurs …
String containing a terminal ID code	The value of the **terminalId** field was retrieved successfully.
Null	No value was associated with the **terminalId** field in the transaction.

Throws

None.

Example

For an example of how the **getTerminalId** method can be used, see the example for **Transaction** on page 4-484.

getTimestamp

Description

The **getTimestamp** method retrieves the value of the **timestamp** field for the transaction. This value represents the date and time the transaction was performed. Use this method to display transaction information.

Syntax

```
public java.sql.Timestamp getTimestamp()
```

Parameters

Parameter	Description
None	

Return Value

The **getTerminalId** method returns the following value:

This value is returned …	if this occurs …
Timestamp	The value of the **timestamp** field was retrieved successfully.

Throws

None.

Example

For an example of how the **getTimestamp** method can be used, see the example for **Transaction** on page 4-484.

getUserId

Description

The **getUserId** method retrieves the value of the **userId** field for the transaction. This value represents the login ID of the user who performed the transaction. Use this method to display transaction information.

Syntax

```
public java.lang.String getUserId()
```

Parameters

Parameter	Description
None	

Return Value

The **getUserId** method returns the following value:

This value is returned …	if this occurs …
String containing a user ID	The value of the **userId** field was retrieved successfully.

Throws

None.

Example

For an example of how the **getUserId** method can be used, see the example for **Transaction** on page 4-484.

toString

Description

The **toString** method provides a string representation of the transaction information, delimited by commas. Use this method to display transaction information.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing transaction data	The transaction information was retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in **com.stc.eIndex.active.core.DataObject**

Example

The following example creates a new Transaction object, **trans**, using parameters supplied by the user application. It then calls the **toString** method to display the transaction information. The following list illustrates the result of calling **toString**.

```
UI,SBYN,A04,SRC,RAD,RD2,null
```

```
...
 Transaction trans = new Transaction(user_id, system, source,
   department, terminal_id);

    System.out.println(trans.toString());
...
```

Uid Class

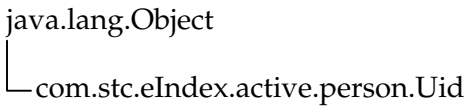
Description

The **Uid** class is a representation of a UID, which is the unique identification code assigned by e*Index.

Properties

The **Uid** class has the following properties:

- Public class
- Implements java.io.Serializable
- Extends **java.lang.Object**



Constructor

The **Uid** class has one constructor, which is described in detail on the page number listed below:

- **Uid** on page 4-494

Fields

The fields included in the **Uid** class are described on the following pages:

- **NULL** on page 4-496

Methods

The method included in the **Uid** class is described in detail on the page number listed below:

- **equals** on page 4-495
- **toString** on page 4-496

Inherited Methods

The **Uid** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

Uid

Description

The **Uid** method is the constructor method for the UID class. Use this method to create objects containing a person's UID information.

Syntax

```
public Uid(java.lang.String uid)
```

or

```
public Uid(long uid)
```

Parameters

Parameter	Description
uid	A unique identification number assigned by e*Index. This parameter can be of the type **long** or **String**.

Return Value

The **Uid** method returns the following value:

This value is returned …	if this occurs …
Uid object	The object representing the UID was created successfully.

Throws

The **Uid** method throws the following exception:

- **EiException Class**

Example

The following example creates a new Person object named **person**, and then uses a UID (specified by the variable *UID_number*) to retrieve the person information to load into **person**. It calls **Uid** to create the Uid object containing the UID to use.

```
...
 Person person = new Person();
 personBO.loadPerson(new Uid(UID_number), person);
...
```

equals

Description

The **equals** method compares two Uid objects to see if they are identical.

Syntax

```
public boolean equals(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The object to compare against.

Return Value

The **equals** method returns one of the following Boolean values:

This value is returned …	if this occurs …
True	The two Uid objects are identical.
False	The two Uid objects are not identical.

Throws

None.

Additional Information

Overrides:

- **equals** in class java.lang.Object (see your Java documentation for more information)

toString

Description

The **toString** method retrieves a string representation of the Uid object.

Syntax

```
public java.lang.String toString()
```

Parameters

Parameter	Description
None	

Return Value

The **toString** method returns the following value:

This value is returned …	if this occurs …
String containing a UID	The string representation was retrieved successfully.

Throws

None.

Additional Information

Overrides:

- **toString** in the **java.lang.Object** class (see your Java documentation for more information about this method)

NULL

Description

The **NULL** field specifies that the field to be set to null is of the type *Uid*.

Syntax

```
public static final Uid NULL
```

ZipCodeLookup Class

Description

The **ZipCodeLookup** class is used to validate zip code values by looking up the city and state information based on the zip code and zip extension.

Properties

The **ZipCodeLookup** class has the following properties:

- Public class
- Extends **java.lang.Object**

```
java.lang.Object
    |
    └─com.stc.eIndex.active.person.ZipCodeLookup
```

Constructor

None.

Methods

The methods included in the **ZipCodeLookup** class are described in detail on the following pages:

- **getCityState** on page 4-498
- **getInstance** on page 4-499

Inherited Methods

The **ZipCodeLookup** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **toString**
- **wait**

getCityState

Description

The **getCityState** method searches the city and state pairs for a given zip code and optionally the zip code extension, and returns a listing of CityState objects. This method can take just a zip code as a parameter or both the zip code and the zip code extension.

Syntax

```
public java.util.Enumeration getCityState(java.lang.String zip)
```

or

```
public java.util.Enumeration getCityState(java.lang.String zip,
java.lang.String zipExt)
```

Parameter

Parameter	Description
zip	The zip code for the city and state you are looking up.
zipExt	The zip code extension for the city you are looking up. This parameter is optional.

Return Value

The **getCityState** method returns one of the following values:

This value is returned …	if this occurs …
Enumeration of CityState values	The zip code lookup successfully retrieved the corresponding city, state, county, and residence code values.
Null	No city or state values were associated with the given zip code.

Throws

The **getCityState** method throws the following exceptions:

- **EiException Class**
- **java.sql.SQLException**

Example

The following example looks up the city, state, county, and residence code associated with the specified zip code and, optionally, the zip code extension. First the sample defines the variables **cs** and **e**, and then gets an instance of EiServer and ZipCodeLookup. It then checks whether a zip code extension was included as a parameter in order to determine which syntax of the call to use, and then calls **zipCodeLookup** to retrieve a list of cities, states, counties,

and residence codes for the specified zip code and zip extension. Finally, the program calls **getCity**, **getState**, **getCounty**, and **getResidenceCode** to display the city, state, county, and residence code associated with the specified zip code and extension.

```
...
 CityState cs;
 Enumeration e = null;
 String zip = zip_code;
 String zipext = zip_code_extension;

   EiServer eiServer = new EiServer("EiServer.properties");
   ZipCodeLookup zipCodeLookup = ZipCodeLookup.getInstance(eiServer);

   if (zipext == "") {
     e = zipCodeLookup.getCityState(zip);
   } else {
     e = zipCodeLookup.getCityState(zip, zipext);
   }

   if (e != null) {
     if (!e.hasMoreElements()) {
       System.out.println("No matching records");
     } else {
       while (e.hasMoreElements()) {
         cs = (CityState)e.nextElement();
         System.out.println("City:  " + cs.getCity());
         System.out.println("State: " + cs.getState());
         System.out.println("County:  + cs.getCounty());
         System.out.println("Residence Code: " + cs.getResidenceCode());
     }
   }
 }
...
```

getInstance

Description

The **getInstance** method retrieves the single instance of the ZipCodeLookup class for the given EiServer object, maintaining a singleton pattern so database results are stored for servicing subsequent zip code lookups.

Syntax

```
public static ZipCodeLookup getInstance(EiServer eiServer)
```

Parameters

Parameter	Description
eiServer	The EiServer object for which the ZipCodeLookup instance is retrieved.

Return Value

The **getInstance** method returns the following value:

This value is returned …	if this occurs …
Instance of ZipCodeLookup	The instance of ZipCodeLookup was retrieve successfully for the specified EiServer object.

Throws

The **getInstance** method throws the following exceptions:

- **java.sql.SQLException**

Additional Information

For more information about the **EiServer Class**, see page 4-319.

Example

To see how **getInstance** in the ZipCodeLookup class can be used, see the example code provided for **getCityState** on page 4-498.

Chapter 5

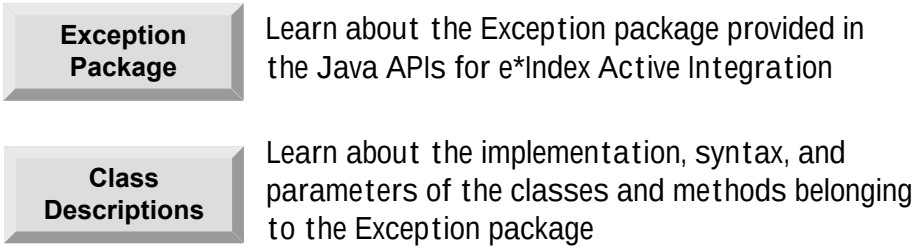
Exception Package

About this Chapter

Overview

This chapter provides a complete listing of Java classes and methods included in the **com.stc.eIndex.active.exception** package for e*Index. It also includes an example of how the parent exception class can be used.

The following diagram illustrates each major topic in this chapter.



About the Exception Package

Overview

This section of the chapter provides the background information about the classes and methods provided in the package **com.stc.eIndex.active.exception**.

The Exception Package

The Exception package includes the classes and methods you need to implement in order to define exception handling and error messages. Classes in this package provide exception handling for many errors, including invalid data, invalid parameters, unique key constrain violations, and so on. The classes in this package all implement **java.io.Serializable**, and are extensions of the **java.lang.Exception** class. For a complete list of the classes, and methods included in the Exception package, see Table 2-2 beginning on page 2-17.

Exception Classes

There are two primary exception classes, **EiException**, and **EiRuntimeException**. The **EiException** class encloses several inner classes that can be used for specific processing errors. When you define an exception, you can also specify an error message to be displayed when the exception is thrown. Many of the Java methods included in the Person package create EiException objects when there is an error in processing.

EiException Class

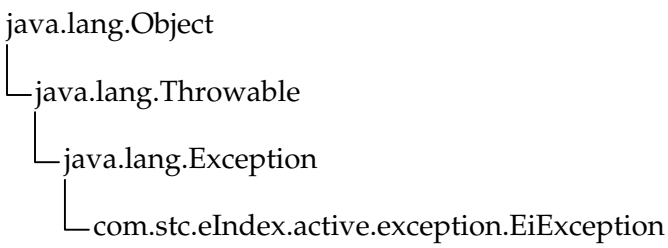
Description

The **EiException** class is the parent exception class for most of the classes in the Exception package.

Properties

The **EiException** class has the following properties:

- Public class
- Implements the **java.io.Serializable** interface
- Direct subclasses include **EiException.DataNotFound**, **EiException.DeleteRow**, **EiException.EmptyString**, **EiException.ExceedMaxStringLength**, **EiException.GeneralException**, **EiException.InvalidData**, **EiException.InvalidHistory**, **EiException.InvalidLocalIdStatus**, **EiException.InvalidParameter**, **EiException.InvalidPersonStatus**, **EiException.InvalidTimestamp**, **EiException.InvalidUid**, **EiException.InvalidUniqueKeyQuery**, **EiException.MatchException**, **EiException.NonUpdateableFieldModified**, **EiException.RequiredFieldIsNull**, **EiException.UniqueKeyException**, and **EiException.UpdateRow**
- Extends **java.lang.Exception**



Constructor

None.

Methods

None.

Inherited Methods

The **EiException** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**

- **getMessage**
- **printStackTrace**
- **toString**

The **EiException** class inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.DataNotFound Class

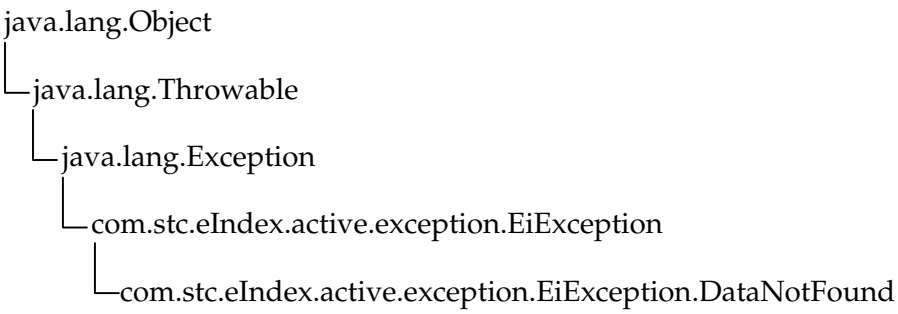
Description

The **EiException.DataNotFound** class represents an exception thrown when required data cannot be found in the database.

Properties

The **EiException.DataNotFound** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.DataNotFound** class has one constructor, which is described on the following page:

- **EiException.DataNotFound** on page 5-7

Methods

None.

Inherited Classes

The **EiException.DataNotFound** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.DataNotFound** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.DataNotFound** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.DataNotFound

Description

The **EiException.DataNotFound** constructor method creates an exception along with the specified error message.

Syntax

```
public EiException.DataNotFound(java.lang.String msg)
```

Parameters

Parameter	Description
msg	A detailed message stating the reason for the exception.

Return Value

The **EiDataException** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.DataNotFound with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.DeleteRow Class

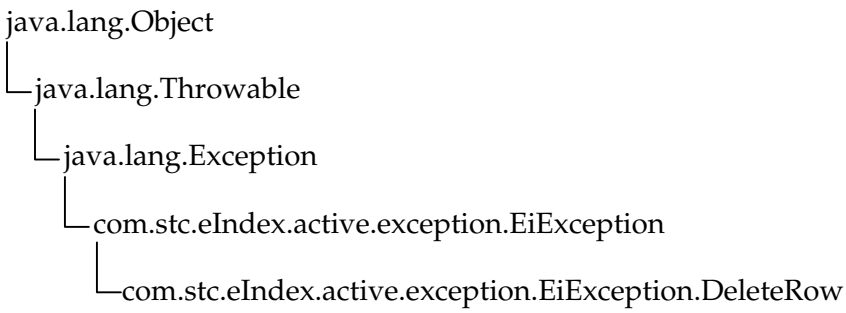
Description

The **EiException.DeleteRow** class represents an exception thrown when a row that is being deleted cannot be deleted. This class is not currently being used.

Properties

The **EiException.DeleteRow** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.DeleteRow** class has one constructor, which is described on the following page:

- **EiException.DeleteRow** on page 5-10

Methods

None.

Inherited Classes

The **EiException.DeleteRow** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.DeleteRow** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.DeleteRow** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.DeleteRow

Description

The **EiException.DeleteRow** constructor method creates an exception along with an error message.

Syntax

```
public EiException.DeleteRow(java.lang.String tableName, int rowCount)
```

Parameters

Parameter	Description
tableName	The name of the table that contains the row causing the exception.
rowCount	The number of rows that were being deleted.

Return Value

The **EiException.DeleteRow** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.DeleteRow with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.EmptyString Class

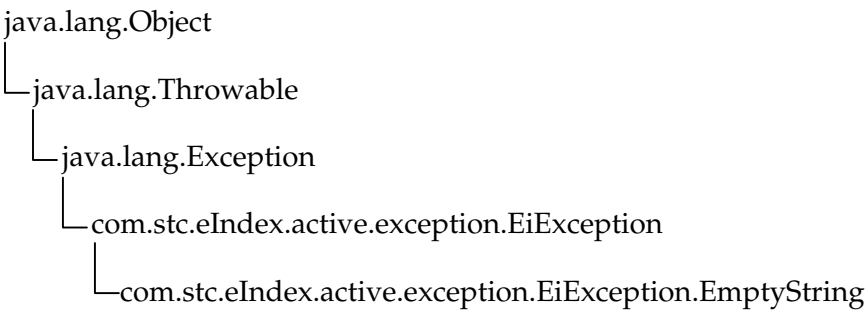
Description

The **EiException.EmptyString** class represents an exception thrown when an empty string is passed into a field that does not allow empty strings.

Properties

The **EiException.EmptyString** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.EmptyString** class has one constructor, which is described on the following page:

- **EiException.EmptyString** on page 5-13

Methods

None.

Inherited Classes

The **EiException.EmptyString** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.EmptyString** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.EmptyString** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.EmptyString

Description

The **EiException.EmptyString** constructor method creates an exception along with an error message.

Syntax

```
public EiException.EmptyString(java.lang.String fieldName,
java.lang.String tableName)
```

Parameters

Parameter	Description
fieldName	The name of the field causing the exception.
tableName	The name of the table that stores the field causing the exception.

Return Value

The **EiException.EmptyString** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.EmptyString with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.ExceedMaxStringLength Class

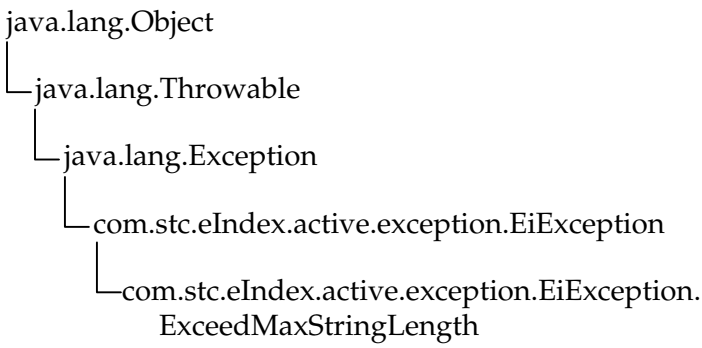
Description

The **EiException.ExceedMaxStringLength** class represents an exception thrown when a value entered into a field is longer than the allowed length for that field.

Properties

The **EiException.ExceedMaxStringLength** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.ExceedMaxStringLength** class has one constructor, which is described on the following page:

- **EiException.** on page 5-19

Methods

None.

Inherited Classes

The **EiException.ExceedMaxStringLength** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.ExceedMaxStringLength** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.ExceedMaxStringLength** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.ExceedMaxStringLength

Description

The **EiException.ExceedMaxStringLength** constructor method creates an exception along with an error message.

Syntax

```
public EiException.ExceedMaxStringLength(java.lang.String
fieldName, java.lang.String tableName, int maxLength,
java.lang.String value)
```

Parameters

Parameter	Description
fieldName	The name of the field causing the exception.
tableName	The name of the table that contains the field causing the exception.
maxLength	The maximum allowed length for the given field.
value	The value that was being entered into the given field and that caused the exception.

Return Value

The **EiException.ExceedMaxStringLength** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.ExceedMaxStringLength with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.GeneralException Class

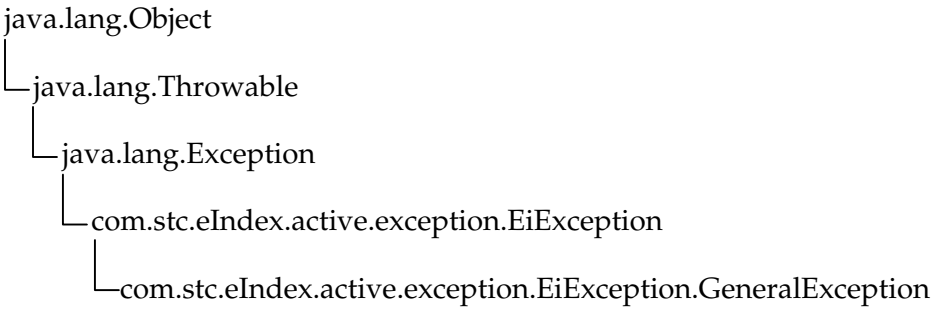
Description

The **EiException.GeneralException** class represents an exception thrown when a general error occurs.

Properties

The **EiException.GeneralException** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.GeneralException** class has one constructor, which is described on the following page:

- **EiException.GeneralException** on page 5-19

Methods

None.

Inherited Classes

The **EiException.GeneralException** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.GeneralException** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.GeneralException** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.GeneralException

Description

The **EiException.GeneralException** constructor method creates an exception along with the specified error message.

Syntax

```
public EiException.GeneralException(java.lang.String msg)
```

or

```
public EiException.GeneralException(Exception e)
```

Parameters

Parameter	Description
msg	The text of the message to display for the general exception.
e	An exception that was already generated. Use this parameter to re-throw an exception that was already generated.

Return Value

The **EiException.GeneralException** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.General Exception with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidData Class

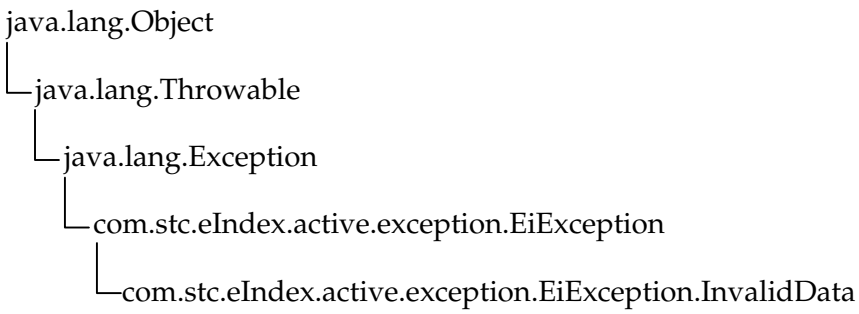
Description

The **EiException.InvalidData** class represents an exception thrown when invalid data is passed into the database.

Properties

The **EiException.InvalidData** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidData** class has one constructor, which is described on the following page:

- **EiException.InvalidData** on page 5-22

Methods

None.

Inherited Classes

The **EiException.InvalidData** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidData** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.GeneralException** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidData

Description

The **EiException.InvalidData** constructor method creates an exception along with the specified error message.

Syntax

```
public EiException.InvalidData(java.lang.String msg)
```

Parameters

Parameter	Description
msg	The text of the message to display for the invalid data exception.

Return Value

The **EiException.InvalidData** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.InvalidData with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidHistory Class

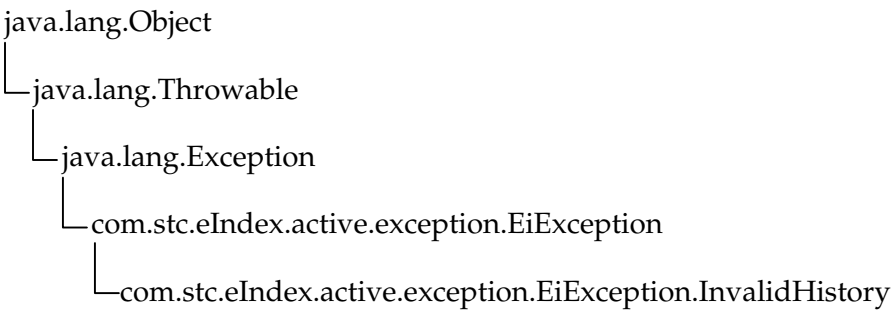
Description

The **EiException.InvalidHistory** class represents an exception thrown when audit trail is invalid. This class is not currently being used.

Properties

The **EiException.InvalidHistory** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidHistory** class has one constructor, which is described on the following page:

- **EiException.InvalidHistory** on page 5-25

Methods

None.

Inherited Classes

The **EiException.InvalidHistory** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidHistory** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidHistory** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidHistory

Description

The **EiException.InvalidHistory** constructor method creates an exception along with the specified error message.

Syntax

```
public EiException.InvalidHistory(java.lang.String tableName,
java.lang.String comment)
```

Parameters

Parameter	Description
tableName	The name of the table in which the invalid information would be stored.
comment	The error message to display.

Return Value

The **EiException.InvalidHistory** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.Invalid History with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidLocalIdStatus Class

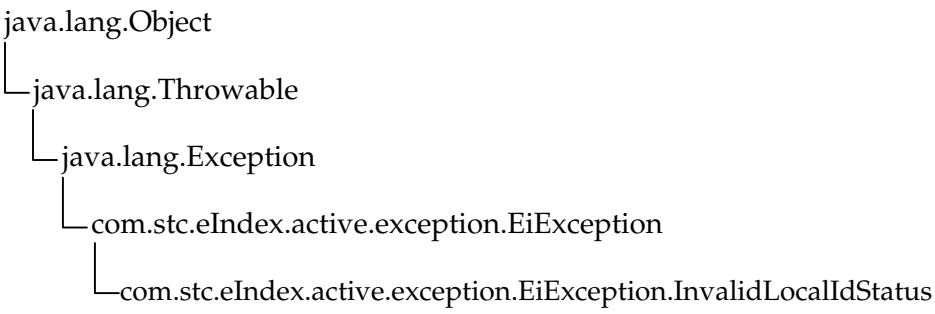
Description

The **EiException.InvalidLocalIdStatus** class represents an exception thrown when the specified local ID status is not valid.

Properties

The **EiException.InvalidLocalIdStatus** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidLocalIdStatus** class has one constructor, which is described on the following page:

- **EiException.InvalidLocalIdStatus** on page 5-28

Methods

None.

Inherited Classes

The **EiException.InvalidLocalIdStatus** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidLocalIdStatus** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidLocalIdStatus** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidLocalIdStatus

Description

The **EiException.InvalidLocalIdStatus** constructor method creates an exception along with an error message.

Syntax

```
public EiException.InvalidLocalIdStatus(java.lang.String code)
```

Parameters

Parameter	Description
code	The local ID status code that was found to be invalid.

Return Value

The **EiException.InvalidLocalIdStatus** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.InvalidLocalIdStatus with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidParameter Class

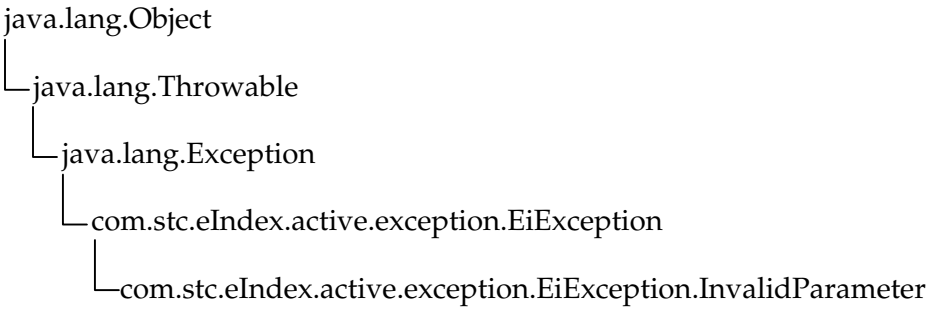
Description

The **EiException.InvalidParameter** class represents an exception thrown when one or more of the parameters passed into an API is not valid.

Properties

The **EiException.InvalidParameter** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidParameter** class has one constructor, which is described on the following page:

- **EiException.InvalidParameter**on page 5-31

Methods

None.

Inherited Classes

The **EiException.InvalidParameter** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidParameter** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidParameter** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidParameter

Description

The **EiException.InvalidParameter** constructor method creates an exception along with the specified error message.

Syntax

```
public EiException.InvalidParameter(java.lang.String comment)
```

Parameters

Parameter	Description
comment	An error message to display when the invalid parameter exception is thrown.

Return Value

The **EiException.InvalidParameter** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.Invalid Parameter with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidPersonStatus Class

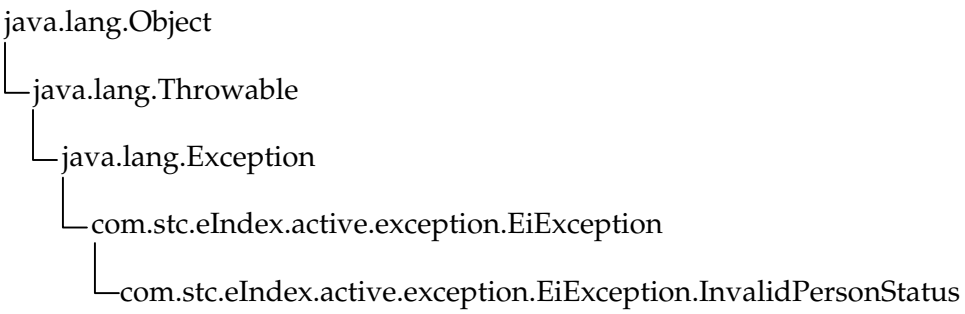
Description

The **EiException.InvalidPersonStatus** class represents an exception thrown when the status specified for a person record is not valid.

Properties

The **EiException.InvalidPersonStatus** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidPersonStatus** class has one constructor, which is described on the following page:

- **EiException.InvalidPersonStatus** on page 5-34

Methods

None.

Inherited Classes

The **EiException.InvalidPersonStatus** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidParameter** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidPersonStatus** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidPersonStatus

Description

The **EiException.InvalidPersonStatus** constructor method creates an exception along with an error message.

Syntax

```
public EiException.InvalidPersonStatus(java.lang.String code)
```

Parameters

Parameter	Description
code	The person status code that caused the invalid person status exception to be thrown.

Return Value

The **EiException.InvalidPersonStatus** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.InvalidPersonStatus with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidTimestamp Class

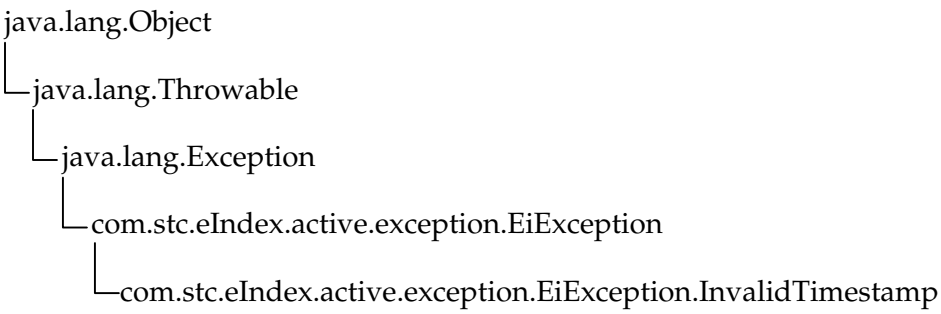
Description

The **EiException.InvalidTimestamp** class represents an exception thrown when a timestamp field is not valid.

Properties

The **EiException.InvalidTimestamp** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidTimestamp** class has one constructor, which is described on the following page:

- **EiException.InvalidTimestamp** on page 5-37

Methods

None.

Inherited Classes

The **EiException.InvalidTimestamp** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidTimestamp** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidTimestamp** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidTimestamp

Description

The **EiException.InvalidTimestamp** constructor method creates an exception along with an error message.

Syntax

```
public EiException.InvalidTimestamp(java.lang.String tableName,
java.lang.String timestamp, java.text.ParseException e)
```

Parameters

Parameter	Description
tableName	The name of the table in which the invalid timestamp is stored.
timestamp	The invalid timestamp value.
e	The ParseException error that was thrown while attempting to parse the timestamp.

Return Value

The **EiException.InvalidTimestamp** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.InvalidTimestamp with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidUid Class

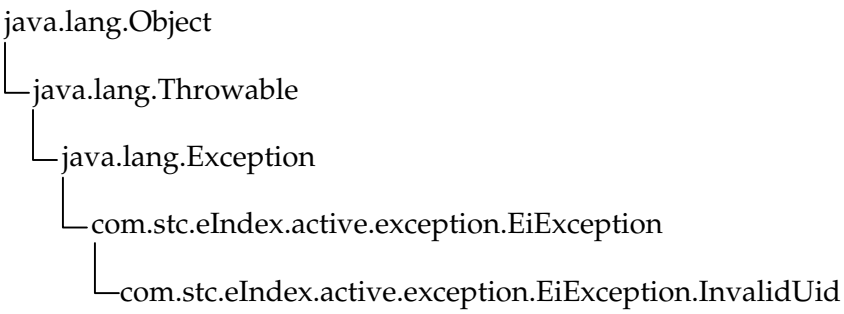
Description

The **EiException.InvalidUid** class represents an exception thrown when the specified UID is not a valid UID.

Properties

The **EiException.InvalidUid** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidUid** class has one constructor, which is described on the following page:

- **EiException.InvalidUid** on page 5-40

Methods

None.

Inherited Classes

The **EiException.InvalidUid** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidUid** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidUid** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidUid

Description

The **EiException.InvalidUid** constructor method creates an exception along with an error message.

Syntax

```
public EiException.InvalidUid(java.lang.String uid)
```

Parameters

Parameter	Description
uid	The UID that is causing the invalid UID exception.

Return Value

The **EiException.InvalidUid** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.**InvalidUid** with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.InvalidUniqueKeyQuery Class

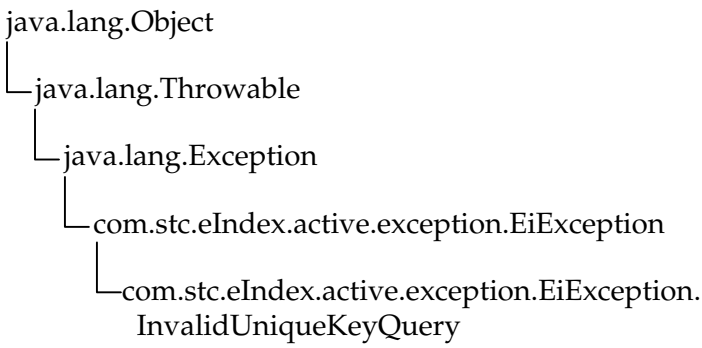
Description

The **EiException.InvalidUniqueKeyQuery** class represents an exception thrown when an invalid value is entered into a unique key field during a query. This class is not currently being used.

Properties

The **EiException.InvalidUid** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.InvalidUniqueKeyQuery** class has one constructor, which is described on the following page:

- **EiException.InvalidUniqueKeyQuery** on page 5-43

Methods

None.

Inherited Classes

The **EiException.InvalidUniqueKeyQuery** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.InvalidUniqueKeyQuery** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.InvalidUniqueKeyQuery** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.InvalidUniqueKeyQuery

Description

The **EiException.InvalidUniqueKeyQuery** constructor method creates an exception along with an error message.

Syntax

```
public EiException.InvalidUniqueKeyQuery(java.lang.String tableName)
```

Parameters

Parameter	Description
tableName	The name of the database table containing the unique key field that was violated.

Return Value

The **EiException.InvalidUniqueKeyQuery** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.InvalidUniqueKeyQuery with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.MatchException Class

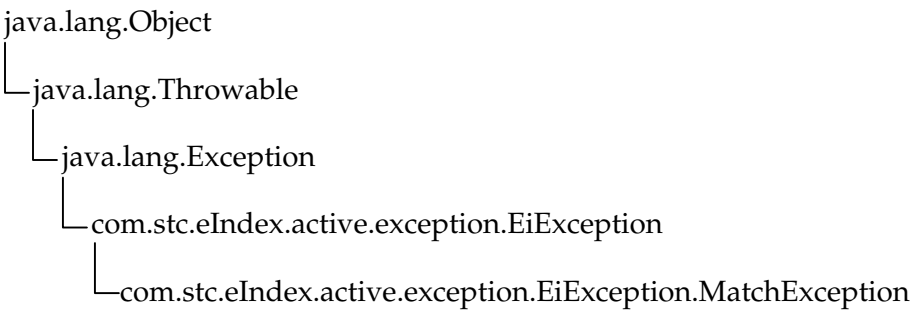
Description

The **EiException.MatchException** class represents an exception thrown when a Vality exception is thrown during the matching process.

Properties

The **EiException.MatchException** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.MatchException** class has one constructor, which is described on the following page:

- **EiException.MatchException** on page 5-46

Methods

None.

Inherited Classes

The **EiException.MatchException** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.MatchException** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.MatchException** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.MatchException

Description

The **EiException.MatchException** constructor method creates an exception along with an error message.

Syntax

```
public EiException.MatchException(com.Vality.jwr.MatchException
e)
```

Parameters

Parameter	Description
e	The Vality exception thrown during the match process.

Return Value

The **EiException.MatchException** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.Match Exception with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.NonUpdateableFieldModified Class

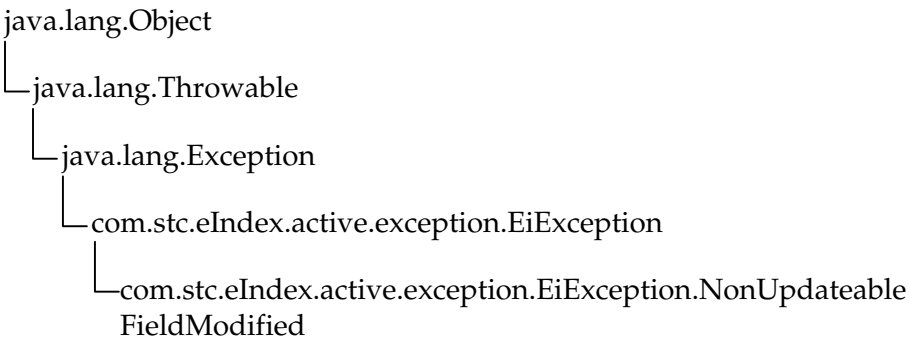
Description

The **EiException.NonUpdateableFieldModified** class represents an exception thrown when the value of a field that cannot be modified is changed during an update.

Properties

The **EiException.NonUpdateableFieldModified** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.NonUpdateableFieldModified** class has one constructor, which is described on the following page:

- **EiException.NonUpdateableFieldModified** on page 5-49

Methods

None.

Inherited Classes

The **EiException.NonUpdateableFieldModified** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.NonUpdateableFieldModified** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.NonUpdateableFieldModified** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.NonUpdateableFieldModified

Description

The **EiException.NonUpdateableFieldModified** constructor method creates an exception along with an error message.

Syntax

```
public EiException.NonUpdateableFieldModified(java.lang.String
databaseColumn, java.lang.String tableName)
```

Parameters

Parameter	Description
databaseColumn	The name of the column in the database that stores the field that cannot be updated.
tableName	The name of the table in which **databaseColumn** is located.

Return Value

The **EiException.NonUpdateableFieldModified** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.Non UpdateableField Modified with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.RequiredFieldIsNull Class

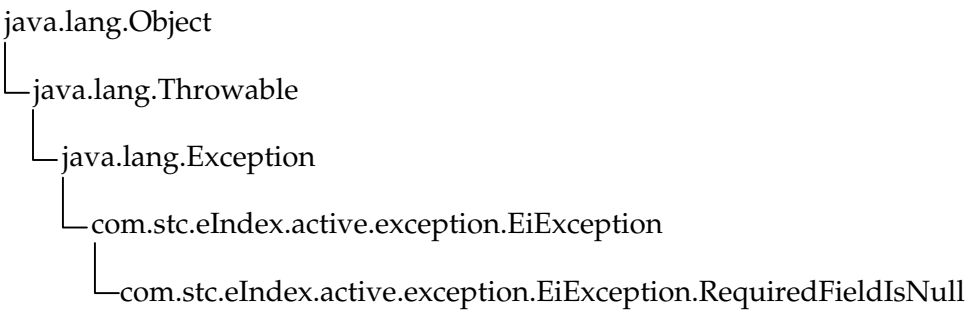
Description

The **EiException.RequiredFieldIsNull** class represents an exception thrown when null is entered into a required field.

Properties

The **EiException.RequiredFieldIsNull** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.RequiredFieldIsNull** class has one constructor, which is described on the following page:

- **EiException.RequiredFieldIsNull** on page 5-52

Methods

None.

Inherited Classes

The **EiException.RequiredFieldIsNull** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.RequiredFieldIsNull** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.RequiredFieldIsNull** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.RequiredFieldIsNull

Description

The **EiException.RequiredFieldIsNull** constructor method creates an exception along with an error message.

Syntax

```
public EiException.RequiredFieldIsNull(java.lang.String
databaseColumn, java.lang.String tableName)
```

Parameters

Parameter	Description
databaseColumn	The name of the column in the database that stores the required field.
tableName	The name of the table in which **databaseColumn** is located.

Return Value

The **EiException.RequiredFieldIsNull** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.RequiredFieldIsNull with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.UniqueKeyException Class

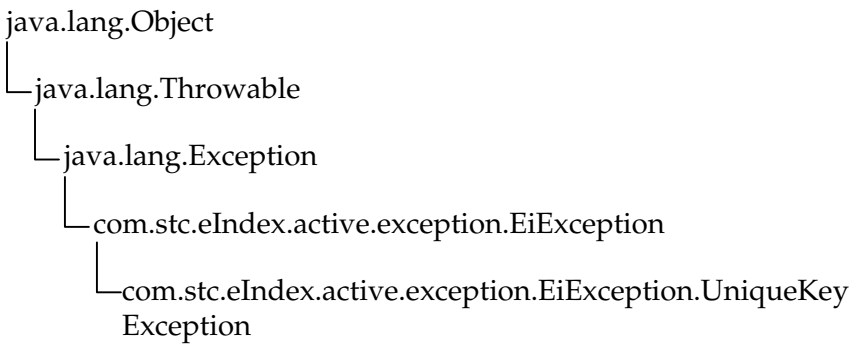
Description

The **EiException.UniqueKeyException** class represents an exception thrown when a unique key constraint is violated during an add or update transaction.

Properties

The **EiException.UniqueKeyException** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.UniqueKeyException** class has one constructor, which is described on the following page:

- **EiException.UniqueKeyException** on page 5-55

Methods

None.

Inherited Classes

The **EiException.UniqueKeyException** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.UniqueKeyException** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.UniqueKeyException** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.UniqueKeyException

Description

The **EiException.UniqueKeyException** constructor method creates an exception along with an error message.

Syntax

```
public EiException.UniqueKeyException(java.lang.String
tableName)
```

Parameters

Parameter	Description
tableName	The name of the table that contains the unique key constraint being violated.

Return Value

The **EiException.UniqueKeyException** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.UniqueKeyException with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiException.UpdateRow Class

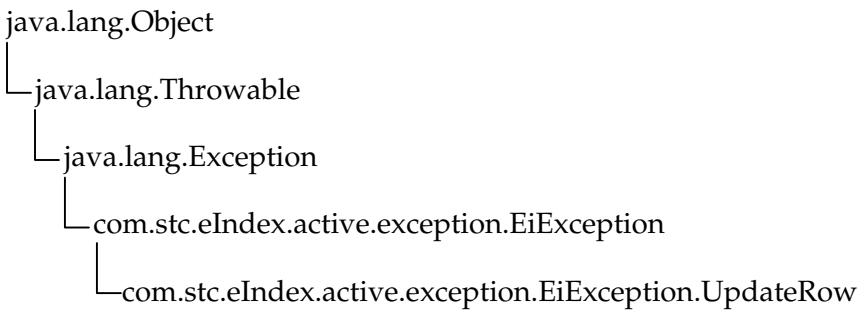
Description

The **EiException.UpdateRow** class represents an exception thrown when a an error occurs while updating a row in the database.

Properties

The **EiException.UpdateRow** class has the following properties:

- Public static class
- Implements the **java.io.Serializable** interface
- Enclosing class is **EiException**
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiException.UpdateRow** class has one constructor, which is described on the following page:

- **EiException.** on page 5-58

Methods

None.

Inherited Classes

The **EiException.UpdateRow** class inherits all of the inner classes in **EiException.**

Inherited Methods

The **EiException.UpdateRow** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiException.UpdateRow** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiException.UpdateRow

Description

The **EiException.UpdateRow** constructor method creates an exception along with an error message.

Syntax

```
public EiException.UpdateRow(java.lang.String tableName, int rowCount)
```

Parameters

Parameter	Description
tableName	The name of the table that contains the row being updated.
rowCount	The number of rows being updated.

Return Value

The **EiException.UpdateRow** method returns the following value:

This value is returned …	if this occurs …
An instance of EiException.UpdateRow with an error message	There was an error in the previous function and an exception was thrown.

Throws

None.

EiRuntimeException Class

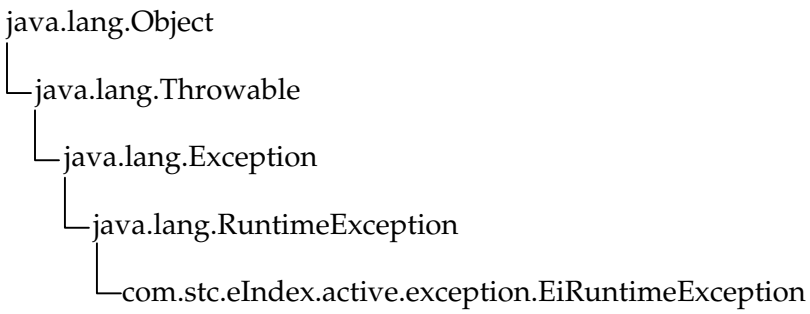
Description

The **EiRuntimeException** class represents a runtime exception thrown during processing.

Properties

The **EiRuntimeException** class has the following properties:

- Public class
- Implements the **java.io.Serializable** interface
- Extends **com.stc.eIndex.active.exception.EiException**



Constructor

The **EiRuntimeException** class has one constructor, which is described on the following page:

- **EiRuntimeException** on page 5-61

Methods

The **EiRuntimeException** class has one method, which is described on the following page:

- **printStackTrace** on page 5-62

Inherited Methods

The **EiRuntimeException** class inherits these methods from **java.lang.Throwable** (see your Java documentation for more information):

- **fillInStackTrace**
- **getLocalizedMessage**
- **getMessage**
- **printStackTrace**
- **toString**

The **EiRuntimeException** class also inherits these methods from **java.lang.Object** (see your Java documentation for more information):

- **equals**
- **getClass**
- **hashCode**
- **notify**
- **notifyAll**
- **wait**

EiRuntimeException

Description

The **EiRuntimeException** constructor method creates a runtime exception along with an error message.

Syntax

```
public EiRuntimeException(java.lang.String msg)
```

or

```
public EiRuntimExcception(java.lang.Exception e)
```

Parameters

Parameter	Description
msg	An error message describing the reason for the exception.
e	The Java exception thrown.

Return Value

The **EiRuntimeException** method returns the following value:

This value is returned …	if this occurs …
An instance of EiRuntimeException with an error message	There was an error in the previous function and a runtime exception was thrown.

Throws

None.

printStackTrace

Description

The **printStackTrace** method displays the exception and its back trace to the specified output.

Syntax

```
public void printStackTrace()
```

Parameters

Parameter	Description
None	

Return Value

The **printStackTrace** method returns the following value:

This value is returned …	if this occurs …
String	The exception and its backtrace were displayed successfully.

Throws

None.

Additional Information

Overrides:

- **printStackTrace** in class java.lang.Throwable (see your Java documentation for more information)