

# ETI Data System Library for JAVA/FS DA

## Release Notes

### Revision 4.3.3A

**December, 2005**

Copyright © 2005 by Evolutionary Technologies International, Inc. All rights reserved.

Evolutionary Technologies International, ETI, ETI Solution, ETI•EXTRACT, the ETI logo, Dialogue Coach, AnswerLink, Success First, and MetaStore are trademarks or registered trademarks of Evolutionary Technologies International, Inc.

Java, the Java JDK, and its related technologies are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries.

All other product and company names mentioned are included for identification only and may be trademarks and/or registered trademarks of their respective companies or institutions.

# Table of Contents

<b>1</b>	<b>PREREQUISITES</b> .....	<b>3</b>
<b>2</b>	<b>DSL INSTALLATION AND LOADING</b> .....	<b>3</b>
	INSTALLING THE DSL ON YOUR HARD DRIVE .....	3
	LOADING THE DSL.....	3
<b>3</b>	<b>RELEASE 4.3.3</b> .....	<b>4</b>
3.1	PURPOSE.....	4
3.2	FEATURES IN RELEASE 4.3.3.....	4
3.2.1	<i>Java language source code generation</i> .....	4
3.2.2	<i>Unified Schema format</i> .....	4
3.2.3	<i>Shell Script Configuration</i> .....	4
3.3	LIMITATIONS OF THIS RELEASE.....	8
3.3.1	<i>Binary Data</i> .....	8
3.3.2	<i>Run Files, Run Parameters, and Key files</i> .....	8
3.3.3	<i>N-Way Merge</i> .....	9
<b>4</b>	<b>RELEASES 4.3.0 – 4.3.2</b> .....	<b>9</b>

---

# 1 PREREQUISITES

---

The DSL for JAVA/FS DA requires the following products to be at the indicated release number *or later*:

- **ETI Solution® Version 5.2.2.** Required to ensure that the DSL will generate the expected code.  
**Note** You must use the Integration Client for the JAVA/FS DA retrieve schema process.
- **Shared Objects 4.3.2.** This component will be loaded automatically during the installation of the DSL when you select the option to auto-load the prerequisites.
- **TCL Functions 4.3.3.** This component will be loaded automatically during the installation of the DSL when you select the option to auto-load the prerequisites.
- **Executive 4.3.1.** This component will be loaded automatically during the installation of the DSL when you select the option to auto-load the prerequisites.
- **JAVA/FS Intermediate Actions 4.3.** Required for supporting enhanced merge processing. This component will be loaded automatically during the installation of the DSL when you select the option to auto-load the prerequisites.

Generated JAVA/FS programs require a Java runtime program and Java compiler so the query, populate, and merge programs can be compiled and executed.

ETI Solution allows the configuration and utilization of any preferred Java runtime or compiler, but is designed for use with Sun Microsystems Java JDK 1.4 or later.

# 2 DSL INSTALLATION AND LOADING

---

## Installing the DSL on your Hard Drive

To install ETI Data System Libraries, you must follow the directions listed in:

- *ETI Solution Administration Guide*, Chapter 2, “Getting Started”

**Warning:** If you do not follow the DSL installation procedures described in the manual listed above, but instead manually copy files from the CD-ROM, then you will not get the updated version of the DSL install script. This will cause the DSL installation to fail.

## Loading the DSL

To load the DSL into a MetaStore, follow the procedure listed in:

- *ETI Solution Administration Guide*, Chapter 3, “Populating MetaStores”

Any available patches will be loaded automatically with the DSL.

## 3 RELEASE 4.3.3

---

### 3.1 Purpose

The primary purpose of this release is to provide support for generating programs to perform data conversions using the Java programming language.

### 3.2 Features in Release 4.3.3

Refer to the **ETI Data System Library DA Procedures** and **ETI Data System Library DA Reference** manuals for information on the standard features all DA DSLs and specific features for the JAVA/FS DA DSL. The following sections provide information specific to the JAVA/FS features and implementation.

#### 3.2.1 Java language source code generation

This DSL generates source code for the Java programming language. While similar in form and function to the C/FS and COBOL/FS DSL's, the JAVA/FS DSL generates object-oriented code to read and/or write Hierarchical data from flat files.

An understanding of the Java programming language is required, and is assumed in most documentation.

#### 3.2.2 Unified Schema format

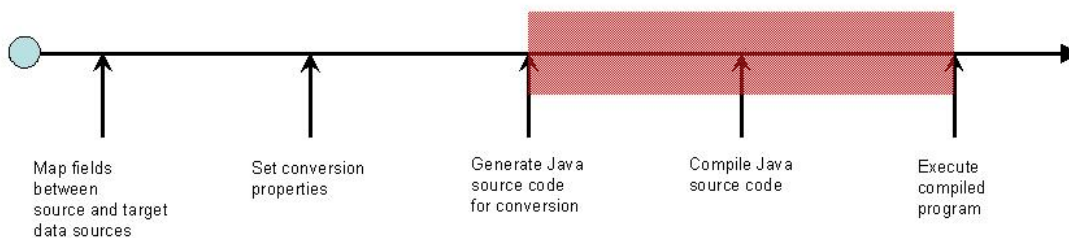
The JAVA/FS DSL utilizes a new format to author schemas for hierarchical data called Unified Retrieve Schema. This schema grammar is in an XML format and allows the most flexibility for authoring schemas which represent hierarchial data. The new schema format also allows the author to include all schema detail in the schema file, and no longer requires the schema author to use the Schema Editor in the ETI Solution client to set data types, or create additional properties.

For more information on Unified Retrieve Schema, see **Chapter 2: Representing Data as an ETI EXTRACT Schema** of the **ETI Data System Library DA Procedures** manual.

#### 3.2.3 Shell Script Configuration

Due to the differing number of computer platforms, Java versions, and Java virtual machines, the JAVA/FS DSL was designed with flexible configuration options to ensure it will work as required in most situations.

Before describing all the configuration options for JAVA/FS, it is important to understand the process of creating conversions using ETI Solution. The following diagram shows the major steps used to create a conversion using ETI Solution:



A conversion is created with a source and target data source. Fields in the conversion source are mapped to the conversion target. Once mappings are complete, any properties specific to the conversion are set and the Java source code is generated.

Once the source code is generated it must be compiled to run as a program. Then the program must be executed to actually perform the data movement defined by the conversion.

This section of the release notes covers the steps in the conversion process which are shaded with red in the diagram above, and how to configure and customize those steps to best integrate with any existing environment.

### 3.2.3.1 Compilation and Execution Properties in JAVA/FS

The following are some of the properties which can be used to configure the compilation and execution of the Java source code generated by ETI Solution:

- `shell_compiler_java` — defines the executable and path to the preferred Java compiler.
- `shell_interpreter_java` — defines the executable and path to the preferred Java interpreter.
- `shell_flags_compiler_java` — defines the command-line options for the executable specified by the `shell_compiler_java` property.
- `java_generate_utils` — controls whether the `ExUtil.java` utilities file is generated. Since these utilities are common to all query and populate programs, they can be generated one time and reused by all subsequent conversions.
- `shell_classpath` — specifies the classpath used for compilation and execution of the generated Java code.
- `java_package_program_name` — controls the naming of the packages and related directories for the generated Java code.
- `java_package_util_name` — controls the naming of the packages for the generated Java utility source files.

**For details on all JAVA/FS properties, their options, and usage, see Chapter 1 “Properties” in the *DA Reference manual*.**

Examples of generated programs that use these properties are provided later in this section along with the default values.

### 3.2.3.2 Understanding Java Packages

The Java programming language introduces the concept of a software package. A package consists of directories of compiled Java source code which are organized into groups of common functionality or usage.

Packages use names that relate to the directory structure of the source code.

Consider the following directory structure and Java source filenames as examples of the package names and related directories for a Java-based card game which is stored in the path `/usr/home/javacode`:

```
/usr/home/javacode/org/acme/games/cards/execute:  
    CardGame.java
```

```
/usr/home/javacode/org/acme/games/cards/card:  
    Card.java  
    Suit.java  
    Value.java
```

```
/usr/home/javacode/org/acme/games/cards/hand:
```

```
    Hand.java
    InHand.java
    Discarded.java

/usr/home/javacode/org/acme/games/cards/deck:
    Cards.java
```

The directory structure is set up to describe the contents of the source code within it.

The `card` directory has the Java files `Card`, `Suit`, and `Value`. These Java files represent everything needed to describe all the facets of a playing card.

Packages in Java source code are described with a period (.) between each directory level. Therefore, the package name of the `card` directory is `org.acme.games.cards.card`. Any Java source files in the `card` directory must have the package declaration of:

```
package org.acme.games.cards.card;
```

There is no particular way that Java packages have to be configured, but it is important to understand that the name of the package and the directory structure of the source code must be similar.

### 3.2.3.3 Package Names Generated from ETI Solution

ETI Solution will generate the following types of Java source code:

- Query program — Generated when the source of a conversion uses the JAVA/FS DSL.
- Populate program — Generated when the target of a conversion uses the JAVA/FS DSL.
- Utility classes — Generated when the source or target of a conversion uses the JAVA/FS DSL.

The default package names for Java source code generated from ETI Solution are described as follows:

- Query program — `com.eti.<conversion_name>.query0`
- Populate program — `com.eti.<conversion_name>.populatel`
- Utility classes — `com.eti.util`

The default package name can be changed by modifying the properties `java_package_program_name` and `java_package_util_name`. See the Chapter 1 “Properties” in the *DA Reference* manual for information on setting these properties.

### 3.2.3.4 Understanding the Java Utility Code

When a conversion is generated, a number of utility functions are created in a file named `ExUtil.java`. These utility functions are the same for any conversion, and the same whether JAVA/FS is used for the source or the target of the conversion.

**NOTE** The utility functions are generated with every conversion, but could be generated one time and then used for all subsequent conversions which are based on JAVA/FS.

In the `ExUtil.java` source code are the definitions of a number of Java classes which have to be placed into specific packages.

To simplify the process of managing packages and directories, part of the build process of the Java code will create the required directories based on the package names and will put the proper Java source files into place.

It is important to understand this concept when configuring the execution host for a conversion.

Using ETI Solution, a conversion is generated as a set of files in a single directory. But the generated code uses package hierarchies that are configurable and require the source code to be placed into specific directories in order to compile and execute.

### 3.2.3.5 Compiling and Executing Java Programs

At the time the Java source code is generated, shell scripts are also generated which will move, compile, and execute the code. A synopsis of these steps are as follows:

1. A conversion is generated from ETI Solution, including the Java source code, scripts to compile and execute the code, and other supporting files and data.
2. A shell script is executed that will create the following directories in the temp area:
  - ./com/eti/util
  - ./com/eti/<conversion\_name>/query0
  - ./com/eti/<conversion\_name>/populate1

**Note** The temp area is defined by a property on the Host object which is used by the Conversion.

3. A script is executed which will read the ExUtil.java file from the conversion directory and split the file into numerous Java source files and place them in the util directory in the temp area.
4. A script is executed which will read the xxxxxxquery0.java file from the conversion directory and split the file into numerous Java source files and place them in the proper directory in the temp area based on the java package name.
5. A shell script calls the compilation process using the version of Java which has been configured.
6. A shell script executes the compiled program to query and/or populate data based on the conversion specification.

### 3.2.3.6 Example: Compiling and Executing a Specific Conversion

To provide a specific example of the compilation and execution of Java source code from an ETI Conversion specification, the following example is provided. The steps are numbered to match the steps in the previous section for reference.

1. A conversion named DB\_JAVAFS is created in the Fred workset. The conversion is in a MetaStore named test\_conv and resides on a UNIX-based system.  
Therefore, the path where all the generated files will reside is as follows:  
/local/MetaStore\_r52/files/test\_conv/System/Fred/cnmdir/DB\_JAVAFS.1
2. The shell script named DB\_JAVAFSquery0.sh is executed that will create the following directories in the temp area /tmp/fred:
  - /tmp/fred/com/eti/util
  - /tmp/fred/com/eti/DB\_JAVAFS/query0
  - /tmp/fred/com/eti/DB\_JAVAFS/populate1

**Note** The temp area is defined by a property on the Host object which defaults to /tmp/[username].

The package names are defined by a property that defaults to com.eti.<conversion\_name>

3. A Perl script named splitClasses.pl is executed from the conversion directory which will read the ExUtil.java file and split the file into numerous Java source files and place them in the /tmp/fred/com/eti/util directory.  
**Note** The generation of the ExUtil.java file and the splitClasses.pl script are specified by the property java\_generate\_utils, which is true by default.
4. A Perl script named splitClasses.pl is executed from the conversion directory which will read the DB\_JAVAFSquery0.java file and split the file into numerous Java source files and place them in the /tmp/fred/com/eti/DB\_JAVAFS/query0 directory.
5. A shell script calls the compilation process using the version of Java which has been configured:  
javac -classpath /tmp/fred /tmp/fred/com/eti/DB\_JAVAFS/query0/\*.java

**Note** The java compiler is specified by a property that defaults to the value javac. Since no path

is specified by default, the `javac` executable would be found in the user's `PATH`.

The classpath is specified by a property that defaults to the user's temporary directory.

6. A shell script executes the compiled program to query and/or populate data based on the conversion specification.

```
java -classpath /tmp/fred -DOUT= . . .  
com.eti.DB_JAVAFS.query0.DB_JAVAFSquery0.java
```

### 3.2.3.7 Guidelines for Conversion Compilation and Execution with JAVA/FS

- Configure the **Temporary Area** in the ETI Solution Host object. This property will determine where files will go for the compilation of the Java source code. Note that this property can be set so that each user of ETI Solution has a separate Temporary Area to work in.
- To ensure the proper files are generated for JAVA/FS, and to ensure that the files compile without error, be sure to configure and select the `Java` and `Perl` languages in the ETI Solution Host object.
- The following characters cannot be used in the name of a conversion using JAVA/FS, or the generated Java source code will not compile properly:

slash (/)	ampersand (&)
hyphen (-)	at (@)
backslash (\)	dollar (\$)
period (.)	brackets ({}[]<>)
colon (:)	parens ( )
semicolon (;)	plus (+)
tilde (~)	exclamation (!)
asterisk (*)	pipe ( )
percent (%)	quotes (" ' `)
pound (#)	caret (^)

- Sun Microsystems Java 1.4.X is recommended as the Java compiler and interpreter. Version 1.3.X is supported for standard Conversions. Use of debug Conversions with the properties `debug_das_msg` and `debug_das_data` require Java version 1.4.X or later due to the use of some API's introduced after Java 1.3.X. Version 5.0 (1.5.X) is supported, but some versions of this Java release have issues which are not yet resolved. Other Java compilers and interpreters can be configured if required.

## 3.3 Limitations of This Release

The following limitations apply to this release of the DSL for JAVA/FS DA.

### 3.3.1 Binary Data

The Java programs generated with this release do not support the reading or writing of binary data.

### 3.3.2 Run Files, Run Parameters, and Key files

Auxiliary files like run files and key files are not supported in this release.

### **3.3.3 N-Way Merge**

The JAVA/FS Intermediate Actions does not yet support the ability to perform n-way merges. An n-way merge can be disabled by setting the property **disable\_nway\_merge** to **true** on the conversion or installation objects.

## **4 RELEASES 4.3.0 – 4.3.2**

---

Not applicable. 4.3.3 is the first Generally Available 4.3 release of the DSL for JAVA/FS DA.