

# ETI Data System Library for C/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.

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>4</b>
<b>2</b>	<b>DSL INSTALLATION</b> .....	<b>4</b>
	INSTALLING THE DSL ON YOUR HARD DRIVE .....	4
	LOADING THE DSL.....	4
<b>3</b>	<b>IMPORTANT NOTICES</b> .....	<b>4</b>
3.1	POTENTIAL IMPACT FOR PREVIOUS DA DSL INSTALLATIONS .....	4
3.1.1	<i>Cleanse Filters and VarChars</i> .....	4
3.1.2	<i>Large Integer and Floating Point Number Support and Temporary Variables</i> .....	5
3.1.3	<i>Changes to the Shell/Bat Templates between 4.2 and 4.3</i> .....	5
<b>4</b>	<b>RELEASE 4.3.3</b> .....	<b>7</b>
4.1	PURPOSE.....	7
4.2	FEATURES IN RELEASE 4.3.3.....	7
4.2.1	<i>Cross-DAS Lookup Support</i> .....	7
4.3	LIMITATIONS OF THIS RELEASE.....	7
<b>5</b>	<b>RELEASE 4.3.2</b> .....	<b>7</b>
5.1	PURPOSE.....	7
5.2	FEATURES IN RELEASE 4.3.2.....	7
5.2.1	<i>Single-Step Processing Now Supported</i> .....	8
5.2.2	<i>Delimited Files Can Now Be Defined as an Array</i> .....	8
5.3	LIMITATIONS OF THIS RELEASE.....	8
5.3.1	<i>Retrieve Schema</i> .....	8
5.3.2	<i>Deprecated Grammar Options in Business Rules</i> .....	8
5.3.3	<i>N-Way Merge Not Yet Supported</i> .....	8
5.3.4	<i>Variable Length Arrays</i> .....	8
5.3.5	<i>Formatting Options for String Numerics and Intervals</i> .....	8
5.3.6	<i>Programmatically Handle Misaligned Data</i> .....	8
5.3.7	<i>Automatically Handle Bytes of Padding in Input Data</i> .....	8
5.3.8	<i>Break Keys Defined on Source Side Disable Single-Step Generation</i> .....	9
<b>6</b>	<b>RELEASE 4.3.1</b> .....	<b>9</b>
6.1	PURPOSE.....	9
6.2	FEATURES IN RELEASE 4.3.1.....	9
6.2.1	<i>Large Integer and Floating Point Number Support</i> .....	9
6.2.2	<i>Aggregation</i> .....	9
6.2.3	<i>Auxiliaries</i> .....	9
6.2.4	<i>In Memory Lookup</i> .....	9
6.2.5	<i>C Union Data Type</i> .....	9
6.2.6	<i>New Property to Specify Slack Bytes in Input Data</i> .....	10
6.2.7	<i>C Intermediate Actions</i> .....	10
6.3	LIMITATIONS OF THIS RELEASE.....	10
6.3.1	<i>Single-Step Processing Not Yet Supported</i> .....	10
6.3.2	<i>Retrieve Schema</i> .....	10
6.3.3	<i>N-Way Merge Not Yet Supported</i> .....	10
6.3.4	<i>Delimited Files Cannot Be Defined as an Array</i> .....	10
6.3.5	<i>Variable Length Arrays</i> .....	10
6.3.6	<i>Formatting Options for String Numerics and Intervals</i> .....	10
6.3.7	<i>Programmatically Handle Misaligned Data</i> .....	10

6.3.8	<i>Automatically Handle Slack Bytes in Input Data</i>	11
<b>7</b>	<b>RELEASE 4.3</b>	<b>11</b>
7.1	PURPOSE	11
7.2	FEATURES IN RELEASE 4.3	11
7.2.1	<i>Flexible File Structure Handling</i>	11
7.2.2	<i>Multiple Record Types</i>	11
7.2.3	<i>Processing Hierarchies</i>	11
7.2.4	<i>Enhanced Dialogs for Entering Arithmetic Expressions in Business Rules</i>	11
7.2.5	<i>Access to Pre and Post Transformed Values</i>	12
7.2.6	<i>New Options for Debugging Input during Program Execution</i>	12
7.2.7	<i>C Intermediate Actions DA 4.3</i>	12
7.2.8	<i>Setting the Driver Unit</i>	12
7.2.9	<i>Modifications to <code>exfcns.c</code> and <code>exfcns.h</code></i>	13
7.3	LIMITATIONS OF THIS RELEASE	13
7.3.1	<i>No support for Single Step Processing</i>	13
7.3.2	<i>Retrieve Schema</i>	13
7.3.3	<i>Aggregation</i>	13
7.3.4	<i>Auxiliaries</i>	13
7.3.5	<i>In Memory Lookup</i>	13
7.3.6	<i>Delimited files cannot be defined as an array</i>	13
7.3.7	<i>C Union data type</i>	14
7.3.8	<i>Variable length arrays</i>	14
7.3.9	<i>Formatting Options for String Numerics and Intervals</i>	14
7.3.10	<i>Programmatically handle misaligned data</i>	14
7.3.11	<i>Automatically handle slack bytes in input data</i>	14

# 1 PREREQUISITES

---

The DSL for C/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 C/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.
- **C 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.

## 2 DSL INSTALLATION

---

### Installing the DSL on your Hard Drive

To install ETI Data System Libraries, 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, which 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”.

## 3 IMPORTANT NOTICES

---

### 3.1 Potential Impact for Previous DA DSL Installations

The DSL for C/FS DA 4.3.3 includes enhancements and fixes that are applicable for all C-based DSLs. These changes impact the generated `exfns.h` and `exfns.c` files, which are not specific to a version of the DSL. **This can result in a change in behavior to existing 4.2 C-based DSLs, if the DSL for C/FS DA is loaded into a MetaStore™ which already contains 4.2 C-based DSLs.**

You should review the changes listed below before loading the DSL for C/FS DA into an existing MetaStore.

#### 3.1.1 Cleanse Filters and VarChars

The DSL for C/FS DA 4.3.2 includes fixes for an error that could occur in prior releases of the C-based DSLs when using cleanse filters and VARCHAR data. Application of a cleanse filter (compress multiple spaces, reformat, etc.) that can remove white space within a varchar may provide a result that contains no

(character) data; that is, its length is zero. Previous releases of the C-based DSLs would incorrectly interpret the result as an empty string (“”), while the applied fixes now correctly interpret the result as null.

### 3.1.2 Large Integer and Floating Point Number Support and Temporary Variables

The DSL for C/FS DA includes enhancements for Large Integer and Floating Point Number support and improvements in the handling of temporary variable allocation which can impact existing 4.2 C-based DSLs.

A set of patches for 4.2-based DSLs are provided with the DSL for C/FS DA 4.3.2 release to provide backward compatibility with the changes. **When loading C/FS DA 4.3.2 into an existing MetaStore containing 4.2 C-based DSLs, the following patches must be loaded to ensure correct behavior of the existing 4.2 DSLs:**

- `rel_c_42_patch.RELC42P_4_2_102_1`
- `core_c_42_patch.COREC42P_4_2_102_1`

As these patches are for the 4.2 release, they will not be automatically loaded with C/FS DA 4.3.2. **Refer to the *ETI Solution Administration Guide*, Chapter 3, “Load DSL Updates”** for instructions on loading these patches into your MetaStore.

Note that when loading into an existing development environment in which you are using a previously-generated `exfcns` object file, you will need to re-create `exfcns.o`. If you normally generate programs with the `ex_generate_static_h` and `ex_generate_static_c` properties set to false, then you must temporarily set these properties to true in order to re-generate the `exfcns` files after loading the DSL for C/FS DA 4.3.2. Once the `exfcns.h` and `exfcns.c` files have been re-generated, you may reset these properties to false and avoid regenerating the `exfcns` files each time you generate programs.

### 3.1.3 Changes to the Shell/Bat Templates between 4.2 and 4.3

DA 4.3 DSLs contain changes to the structure of the Shell/Bat templates. These changes provide a common set of Template Definitions that can be extended by attaching DSL-specific Template Libraries.

DA 4.2 DSLs provided DSL-specific Template Definitions for the shell/bat scripts, each of which contained one or two DSL-specific Template Libraries and a large number of DSL-independent (core) libraries. For example, C/DB2 DA 4.2 contained the following Template Definitions:

```
cdb2_sh_qry_cmp_da, cdb2_sh_qry_exe_da, cdb2_sh_qry_std_da
cdb2_sh_pop_cmp_da, cdb2_sh_pop_exe_da, cdb2_sh_pop_std_da
cdb2_bat_qry_cmp_da, cdb2_bat_qry_exe_da, cdb2_bat_qry_std_da
cdb2_bat_pop_cmp_da, cdb2_bat_pop_exe_da, cdb2_bat_pop_std_da
```

The DA 4.3 DSLs provide a common set of shell/bat Template Definitions for all DSLs:

```
sh_qry_cmp_da, sh_qry_exe_da, sh_qry_std_da
sh_pop_cmp_da, sh_pop_exe_da, sh_pop_std_da
bat_qry_cmp_da, bat_qry_exe_da, bat_qry_std_da
bat_pop_cmp_da, bat_pop_exe_da, bat_pop_std_da
```

By default, the 4.3 Template Definitions contain common DSL-independent (core) libraries. However, at DSL load time, DSL-specific Template Libraries are attached to the Template Definitions, providing the functionality specific to that DSL.

For example, when loading a MetaStore with the C/FS DSL, the `sh_cfs_43` Template Library will be added to the `sh_pop_cmp_da` Template Definition to provide functionality specific to C/FS DA.

Subsequent loading of the C/DB2 DA DSL into the same MetaStore will add sh\_cdb2\_43 to the sh\_pop\_cmp\_da Template Definition and provide the C/DB2 DA functionality.

## 4 RELEASE 4.3.3

---

### 4.1 Purpose

The primary purpose of this release is to add new functionality to the DSL.

### 4.2 Features in Release 4.3.3

This release provides the ability to perform cross-das lookups between C/FS DA and other 4.3.3 DA DSLs; for example C/DB2 DA 4.3.3, C/ORACLE DA 4.3.3, etc. The C/FS DA DSL, as well as the DSL that you will be performing the lookup with, must be loaded into the MetaStore. The Sections below provide information on the cross-das lookup functionality provided within this release.

#### 4.2.1 Cross-DAS Lookup Support

The C/FS DA 4.3.3 DSL can be used with the C relational DA 4.3.3 DSLs like C/DB2 and C/Oracle to create conversions with the following types of lookups:

- C/Relational to C/FS Query In-Memory Lookup
- C/Relational to C/FS Populate In-Memory Lookup
- C/FS to C/Relational Query In-Memory Lookup
- C/FS to C/Relational Query Database Lookup
- C/FS to C/Relational Populate Database Lookup
- C/FS to C/Relational Populate In-Memory Lookup

Refer to the release notes for the specific 4.3.3 C/Relational DSL for instructions on how to extend the Query and Populate Templates to provide the cross-das lookup functionality.

### 4.3 Limitations of This Release

The 4.3.3 Release of C/FS DA DSL contains the same limitations as contained in the 4.3.2 release.

## 5 RELEASE 4.3.2

---

### 5.1 Purpose

The primary purpose of this release is to remove limitations and correct errors found in prior releases. A complete list of CRs (software problems) fixed in this release can be provided by the AnswerLine upon request.

### 5.2 Features in Release 4.3.2

The following features have been added in Release 4.3.2, many removing previous limitations in Release 4.3.1.

### 5.2.1 Single-Step Processing Now Supported

By default, optimization level 4 with the current release of the DSL for C/FS DA will generate a single-step conversion if possible. For information about conditions that can prevent generation of single-step programs, refer to Chapter 15, “Single-Step and Multi-Step Conversions”, of the *ETI Data System Library DA: Procedures* manual.

### 5.2.2 Delimited Files Can Now Be Defined as an Array

A file with part delimiters may now be defined with an array such as:

```
char *addressline[3];
```

Note that when defining an array for delimited files, you must now set the property `cfs_parse_subunit_flat` to true in order for the data to be processed correctly.

## 5.3 Limitations of This Release

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

### 5.3.1 Retrieve Schema

The retrieve schema processor will return a parser error if the header file contains a typedef of an array such as:

```
typedef int my_array[12];
```

### 5.3.2 Deprecated Grammar Options in Business Rules

Release 4.3 of all DA DSLs includes updates to grammars to provide for enhanced functionality (see Sections 7.2.4 and 7.2.5 below). As a part of this update, some grammar options are no longer useful or needed. These grammar options are identified in the options pane of the Business Rule Editor as deprecated and will be removed in a future release.

Deprecated options should not be chosen when building new grammars.

### 5.3.3 N-Way Merge Not Yet Supported

The C Intermediate Actions does not yet support the ability to perform n-way merges.

### 5.3.4 Variable Length Arrays

Variable length arrays in which the length of the array is dependent on the value of another part (COBOL OCCURS DEPENDING ON) are not supported.

### 5.3.5 Formatting Options for String Numerics and Intervals

The DSL for C/FS DA currently provides no mechanism for defining how output string parts should be formatted.

### 5.3.6 Programmatically Handle Misaligned Data

Misaligned data (not on the boundaries required by the hardware platform) may cause an addressing exception.

### 5.3.7 Automatically Handle Bytes of Padding in Input Data

The retrieve schema process does not automatically detect the number of bytes of padding (also referred to as *slack bytes*) typically found between characters and integers contained within a structure or union for a given computer's alignment requirements. The property `part_das_pad_length` must be manually set

by the user after retrieving the schema to specify the bytes of padding. Refer to the documentation for this property in Chapter 2, “Representing Data as an ETI•EXTRACT Schema”, of the *DA Procedures* manual for details.

### 5.3.8 Break Keys Defined on Source Side Disable Single-Step Generation

Defining break keys on the source side will force the generation of a multi-step conversion instead of a single-step conversion.

## 6 RELEASE 4.3.1

---

### 6.1 Purpose

The primary purpose of this release is to remove limitations and correct errors found in Release 4.3.0. A complete list of CRs (software problems) fixed in this release can be provided by the AnswerLine upon request.

### 6.2 Features in Release 4.3.1

The following features have been added in Release 4.3.1, many removing previous limitations in Release 4.3.

#### 6.2.1 Large Integer and Floating Point Number Support

Large integers and floating point numbers are now supported with this release. You can now specify the maximum size of an integer and/or floating point number supported by your compiler and conversion using the new properties **c\_bigint\_type** (for integers) and **c\_bigfloat\_type** (for floating point numbers). ETI uses this information to determine the size of the internal storage required. The storage is allocated in the **exfns** static functions.

For details on the values available for the new properties, refer to the documentation in Chapter 1, "Properties", of the *ETI Data System Library DA: Reference* manual.

#### 6.2.2 Aggregation

Aggregation functions are now supported. They are available in a separate aggregation stage in both the query and the populate programs. Refer to Chapter 13, “Aggregating Data”, of the *DA Procedures* manual for details.

#### 6.2.3 Auxiliaries

Auxiliary file support is now provided for run files and run parameters. Refer to Chapter 19, “Key Files, Run Files, and Run Parameters”, of the *DA Procedures* manual for details. Also a new option (**run\_env**) has been added to identify an environment variable as an input parameter. See the documentation for the property **db\_process\_type** in Chapter 1, “Properties”, in the *DA Reference* manual for details.

#### 6.2.4 In Memory Lookup

Lookup operations are now supported. Refer to Chapter 12, “Looking up Data in Auxiliary Sources”, of the *DA Procedures* manual for details.

#### 6.2.5 C Union Data Type

The C union data type is now supported.

### **6.2.6 New Property to Specify Slack Bytes in Input Data**

A new schema property **part\_das\_pad\_length** is available to specify slack bytes contained within the file being read. ETI uses this property to determine how many bytes of padding it should discard when parsing the data for this part from the input file. Refer to the documentation for this property in Chapter 2, “Representing Data as an ETI•EXTRACT Schema”, of the *DA Procedures* manual for details.

### **6.2.7 C Intermediate Actions**

C Intermediate Actions is now functionally equivalent to the COBOL Intermediate Actions except for the ability to perform n-way merges (see section 6.3.3).

## **6.3 Limitations of This Release**

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

### **6.3.1 Single-Step Processing Not Yet Supported**

Optimization level 4 with the current release of the DSL for C/FS DA does not yet support single-step processing. By default, a multi-step conversion with separate query and populate programs and ifiles will be generated. Any merge will be included within the query program.

### **6.3.2 Retrieve Schema**

The retrieve schema processor will return a parser error if the header file contains a typedef of an array such as:

```
typedef int my_array[12];
```

### **6.3.3 N-Way Merge Not Yet Supported**

The C Intermediate Actions does not yet support the ability to perform n-way merges.

### **6.3.4 Delimited Files Cannot Be Defined as an Array**

A file with part delimiters may not be defined with an array such as:

```
char *addressline[3];
```

Instead define it as follows:

```
char *addressline1;  
char *addressline2;  
char *addressline3;
```

### **6.3.5 Variable Length Arrays**

Variable length arrays in which the length of the array is dependent on the value of another part (COBOL OCCURS DEPENDING ON) are not supported.

### **6.3.6 Formatting Options for String Numerics and Intervals**

The DSL for C/FS DA currently provides no mechanism for defining how output string parts should be formatted.

### **6.3.7 Programmatically Handle Misaligned Data**

Misaligned data (not on the boundaries required by the hardware platform) may cause an addressing exception.

### **6.3.8 Automatically Handle Slack Bytes in Input Data**

Slack bytes contained within the file being read must be explicitly declared in the schema. The DSL for C/FS DA assumes there are no slack bytes.

## **7 RELEASE 4.3**

---

### **7.1 Purpose**

The primary purpose of this release is to bring the DSL for C/FS to the DSL Architecture (DA), and to provide hierarchical C file processing support similar to that currently provided by the DSL for COBOL/FS DA.

### **7.2 Features in Release 4.3**

#### **7.2.1 Flexible File Structure Handling**

The DSL for C/FS DA 4.3 provides the ability to read and write almost any file structure: data streams, variable-length files, or fixed-length files. A data stream is a series of ASCII characters that must be parsed into records, units, and parts. Variable-length and fixed-length files are similar to COBOL flat files. The individual parts are fixed in length, but the record may be variable or fixed-length. Fixed and variable-length files may contain binary data, such as an *int*. When reading a data stream, several parsing routines and structures are generated based on the set of delimiters and enclosures defined on the schema. These data delimiters and enclosures may be of any length and value, and enclosures may be required or optional. The parsing routines use this information to determine the record type, when reading multiple record type files, and determining the value of each part in the record. If necessary, the parsing can be fine-tuned by setting properties to control the parsing algorithm, defining multiple delimiters and enclosures, and defining unit or part specific delimiters and enclosures.

#### **7.2.2 Multiple Record Types**

Files containing different record types can be identified and processed with the DSL for C/FS DA. Multiple methods are available for identifying each record type including: value, hierarchy definition, schema position, and value and position. When parsing data streams, the parsing mechanism works with the MRT properties to help determine the type of record read.

#### **7.2.3 Processing Hierarchies**

The process flow for C/FS DA conversions is governed by process hierarchies. The ETI Solution identifies one or more processing hierarchies based on the relationships (joins) and mappings defined in the conversion. Processing hierarchies are the foundation that gives ETI Solution hierarchical DSLs the ability to process complex data structures in a minimum number of instructions.

#### **7.2.4 Enhanced Dialogs for Entering Arithmetic Expressions in Business Rules**

Release 4.3 of all DA DSLs includes a new dialog for entering arithmetic expressions. The new dialog is more streamlined and intuitive than the previous one. The previous dialog is still available to allow replaying business rules defined with the old dialog.

### 7.2.5 Access to Pre and Post Transformed Values

Customers have requested the ability to create a virtual part, define a business rule on the virtual part to calculate a value, and then use this virtual part as input into other business rules in the same processing stage. The problem in the past has been that references to other parts always used the value of the part *before* any business rule was applied. Beginning with DA 4.3, the business rule dialog provides the ability to reference the value of a part as it was before the current processing stage (pre-stage value) or after the current processing stage (post-stage value).

### 7.2.6 New Options for Debugging Input during Program Execution

Two new properties are provided to enable input debugging during program execution:

- `debug_das_msg` — when set to true, enables writing of the new 700 series messages to the log file. These messages provide detailed debugging information for the input read.
- `debug_das_io` — when set to true, enables writing the input data that could not be processed to the log file.

See the “Properties” chapter in the *DA Reference* manual for documentation on these two properties.

### 7.2.7 C Intermediate Actions DA 4.3

Release 5.1 of ETI Solution and C Intermediate Actions DA 4.3 (C/IA DA 4.3) provide the user much better control on how data is merged than in previous releases of the DSL and Intermediate Actions. (For details please refer to the sections beginning with “Merge Processing” in Chapter 16 of the *ETI Data System Library DA: Procedures* manual.)

#### 7.2.7.1 Differences in Merge Processing in C Intermediate Actions DA 4.3

Given the enhanced merge control now provided, the merge processing now differs from the merge in prior releases of C Intermediate Actions DA, which means that existing conversions may exhibit different behavior when the DSL for C/FS DA is installed:

Prior to 5.0, users controlled the order of processing joins through the definition of a driver unit. The driver was specified for the target unit that was populated from a merge. The driver unit determined both the sequence in which the merges were performed, and which join unit would be used to drive each individual merge operation.

The rule of thumb in C Intermediate Actions DA 4.3 is that the “**To**” (child/many) unit is the **driver in each merge operation**. When multiple merge operations are required, the driver unit can change from merge to merge. Therefore, ETI™ recommends that you *not select a driver unit* when using C Intermediate Actions DA 4.3 or later and that you **use the join sequence numbers and the child unit in the relationship as the driver instead** (see “Driver Unit Specified for the Target Unit (Not Recommended)” in Chapter 16 of the *ETI Data System Library DA: Procedures* manual.). You have more control over the order of processing joins using these features.

### 7.2.8 Setting the Driver Unit

Setting the driver unit is typically no longer necessary. For details please refer to the sections “Driver Unit Specified for the Target Unit (Not Recommended)” and “Driver Unit Specified for a Relationship (Rarely Necessary)” in Chapter 16 “Data Relationships and Intermediate Actions” of the *ETI Data System Library DA: Procedures* manual.

## 7.2.9 Modifications to `exfcns.c` and `exfcns.h`

### 7.2.9.1 Type Support

Support routines have been added to `exfcns.c` for unsigned parts.

Support for customizing numbers with integer and floating point data types has been added to the templates that generate `exfcns.c` and `exfcns.h`. This will allow non-standard types such as `_int64` and `long long` to be used to represent integer data types for 64-bit, 128-bit, or longer integers and `extended double`, `long long double` to represent 80-bit, 128-bit, or longer floating point numbers. Support for non-ANSI data-types is highly specific to the processor, compiler, and operating system. As a result, ETI Answerline cannot provide support for non-standard data-types. The Integration Architect should work with the systems' managers to determine if non-standard data-types or the use of compiler switches is the best choice for customizing the installation.

### 7.2.9.2 Performance Enhancements

Several of the routines in `exfcns.c` have been updated, which should result in a slight performance improvement if many type (e.g. integer to string, string to float) transformations are being performed by the DSL.

## 7.3 Limitations of This Release

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

### 7.3.1 No support for Single Step Processing

The current release only supports two-step processing using optimization level 4.

### 7.3.2 Retrieve Schema

The retrieve schema processor will return a parser error if the header file contains a typedef of an array such as:

```
typedef int my_array[12];
```

You cannot have two structs in the C header file with the same variable name (identifier). The resulting schema will not be valid.

### 7.3.3 Aggregation

Aggregation functions are not supported.

### 7.3.4 Auxiliaries

Auxiliary files are not supported.

### 7.3.5 In Memory Lookup

Lookup operations are not supported.

### 7.3.6 Delimited files cannot be defined as an array

A file with part delimiters may not be defined with an array such as:

```
char *addressline[3];
```

Instead define it as follows:

```
char *addressline1;  
char *addressline2;
```

```
char *addressline3;
```

### **7.3.7 C Union data type**

The C union data type is not supported.

### **7.3.8 Variable length arrays**

Variable length arrays in which the length of the array is dependent on the value of another part (COBOL OCCURS DEPENDING ON) are not supported.

### **7.3.9 Formatting Options for String Numerics and Intervals**

The DSL for C/FS DA currently provides no mechanism for defining how output string parts should be formatted.

### **7.3.10 Programmatically handle misaligned data**

Misaligned data (not on the boundaries required by the hardware platform) may cause an addressing exception.

### **7.3.11 Automatically handle slack bytes in input data**

Slack bytes contained within the file being read must be explicitly declared in the schema. The DSL for C/FS DA assumes there are no slack bytes.