



# **ETI • EXTRACT<sup>®</sup>**

---

*Data System Library for C/File System*

*Release Notes*

*Release 1.1.5*

*September, 1998*

Release 1.1.5 — September, 1998

THIS DOCUMENT IS THE CONFIDENTIAL AND PROPRIETARY PRODUCT OF EVOLUTIONARY TECHNOLOGIES INTERNATIONAL, INC. ANY UNAUTHORIZED USE, REPRODUCTION, OR TRANSFER OF THIS DOCUMENT IS STRICTLY PROHIBITED. COPYRIGHT © 1998 BY EVOLUTIONARY TECHNOLOGIES INTERNATIONAL, INC. (SUBJECT TO LIMITED DISTRIBUTION AND RESTRICTED DISCLOSURE ONLY.) ALL RIGHTS RESERVED.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Evolutionary Technologies International, Inc.

#### RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013 (48 CFR Ch.2) and/or the Commercial Computer Software Restricted Rights clause at FAR 52.227.19(c). Unpublished: Rights reserved under the copyright laws of the United States. Contractor/manufacturer is

Evolutionary Technologies International, Inc.  
4301 Westbank Drive, Bldg. B  
Austin, Texas 78746  
(512) 327-6994

<http://www.eti.com>

ETI•EXTRACT and the ETI logo are registered trademarks and Data System Library is a trademark of Evolutionary Technologies International, Inc.

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

— *Proprietary and Confidential* —

Property of  
**Evolutionary Technologies International, Inc.**



# Contents

<b>Introduction .....</b>	<b>1</b>
<i>Contacting ETI .....</i>	<i>1</i>
<b>Related Documentation .....</b>	<b>1</b>
<i>ETI•EXTRACT Online Documentation .....</i>	<i>2</i>
<i>Tcl Man Pages .....</i>	<i>2</i>
<b>Summary of Enhancements and Updates for Release 1.1.3.....</b>	<b>2</b>
<i>Special Character Handling.....</i>	<i>2</i>
<i>Remove/Convert Characters .....</i>	<i>3</i>
<i>Default Value Initialization .....</i>	<i>4</i>
<i>Numeric Format Support.....</i>	<i>4</i>
<i>Separate Sign Field.....</i>	<i>4</i>
<i>Placing Number In Requested Form.....</i>	<i>5</i>
<i>Space fill or Zero fill.....</i>	<i>5</i>
<i>Negative number with parentheses .....</i>	<i>6</i>
<i>Show an explicit sign.....</i>	<i>6</i>
<i>Left Justify .....</i>	<i>7</i>
<i>Right Justify.....</i>	<i>7</i>
<i>Date Format Support.....</i>	<i>7</i>
<i>Date Format Conversion and Validation .....</i>	<i>8</i>
<i>Input Date Formats.....</i>	<i>8</i>
<i>Target-side virtual parts.....</i>	<i>10</i>
<i>PDPR Reported Problems Corrected in Release 1.1.3.....</i>	<i>11</i>
<i>Additions and Changes to Template Functions and Grammars in Release 1.1.3</i>	<i>13</i>
<i>Handbook Updates for release 1.1.3.....</i>	<i>15</i>
<b>Enhancements in Release 1.1.2.....</b>	<b>20</b>

Single Step Processing .....	21
Variable Length String Comparisons.....	21
Content- Sensitive Cleansing/ Translation .....	21
Custom Function Calls .....	22
Internal Link .....	22
External Link .....	23
Special Character Handling .....	24
Remove/Convert Characters.....	24
Substring Functions.....	25
Numeric Format Support .....	25
Numeric Data Cleansing .....	25
Placing Number in Requested Form .....	26
Space fill or Zero fill numeric .....	26
Left Justify.....	26
Date Format Support .....	27
Date Format Conversion and Validation.....	27
Input Date Formats .....	27
Output Date Formats.....	27
Date Arithmetic and Comparisons .....	28
Comparison of two dates .....	28
Add/Subtract days from date.....	28
Difference between two dates.....	29
Use System Date as Default.....	29
Use a Specified Value: Run Date, Current Date, Specified Date .....	29
Aggregation .....	29
Planning the Conversion.....	30
Aggregation Mapping.....	31
Setting Up the Auxiliary Target Databases.....	32
Dimension Parts and Break Keys .....	34
Applying the Aggregation Filter.....	37
Handbook updates for Release 1.1.2 .....	38
Reported Problems Corrected in Release 1.1.2 .....	43

<i>Additions and Changes to Template functions in Release 1.1.2</i> .....	46
<b>Summary of Changes in Release 1.1.1</b> .....	<b>60</b>
<i>Known Limitations of the DSL for C/File System</i> .....	62
<i>Conversion Annotation Field</i> .....	62
<i>Input Data File</i> .....	62
<i>Reading Input Data</i> .....	62
<i>Differences Between Source and Target Schemas</i> .....	63
<i>Multiple Record Type Processing</i> .....	63
<i>Decimals</i> .....	63
<i>C Intermediate Actions</i> .....	63
<i>Writing Integer Data</i> .....	63

# Release Notes

---

## Introduction

These *Release Notes* provide information about Release 1.1.5 of the ETI•EXTRACT Data System Library (DSL) for C/File System.

---

## Contacting ETI

Should you encounter difficulty in using the DSL for C/File System, have questions regarding the documentation, or have feedback or suggestions that can help us improve our product, please contact the ETI Answerline. ETI places a high value on your success and fulfilling your needs as a customer.

You can reach ETI AnswerLine personnel at:

Customer Area	Phone	Fax	E-mail
North America	800-856-0416 (toll free) 512-327-6994 ext. 6660	512-327-6117	eti.answerline@eti.com
United Kingdom	0118-977-1221	0118-977-9800	eti.answerline@eti.com
Europe (outside U.K.)	+44 (0) 118-977-9811	+44 (0) 118-977-9811	eti.answerline-europe@eti.com

ETI's World Wide Web address is <http://www.eti.com>. You can access the AnswerLink™, ETI's online support center, from there.

---

## Related Documentation

Refer to the following documents for additional information:

- Conversion Specialist's Guide*
- Master User's Guide*
- Reference Manual*
- ETI•EXTRACT Data System Library Installation Guide*

**Note:** A README document may be installed with the other Portable Document Format (PDF) files for the DSL for C/FS. If so, the README document contains information that became available after these *Release Notes* were printed.

---

ETI•EXTRACT  
Online  
Documentation

All of the ETI•EXTRACT documentation (including this manual) is available online in the form of Adobe™ Acrobat™ Reader PDF files. They are located in the **doc** subdirectory of your ETI•EXTRACT installation directory. ETI•EXTRACT includes a copy of the UNIX version of Acrobat Reader. To display a list of the available manuals, enter the following command at the UNIX command line:

```
ex_manual
```

To display a particular manual, use the following command:

```
ex_manual manual-name
```

Substitute the name of a manual for *manual-name*.

---

Tcl Man Pages

You can extend the capabilities of ETI•EXTRACT by writing Program Generator Extension Functions and Grammar Extension Functions in Tool Command Language (Tcl). Tcl is a publicly available interpreted language and is embedded in ETI•EXTRACT. For more information about Tcl, refer to *Practical Programming in Tcl and Tk*, Brent B. Welch, ISBN 0-13-182007-9.

ETI•EXTRACT includes UNIX manual (man) pages for Tcl. You can view them by entering the following command at the UNIX command line:

```
man command-name
```

Substitute a Tcl command for *command-name*.

For example, you can display a summary of Tcl language syntax by entering the following command:

```
man Tcl
```

---

Summary of  
Enhancements  
and Updates  
for Release  
1.1.3

This section provides a summary of the additional DSL features, updates, and handbook updates of Release 1.1.3 of the DSL for C/FS.

---

Special  
Character  
Handling

Target side *Modify Value* filters are available for source part translations to convert or remove characters(s) and return a substring of source parts(s).

## Remove/Convert Characters

A Target part Modify Value filter is applied to specify removal/translation of characters from the value of the Source field. This function applies to string extract parts only.

The Convert Characters filter substitutes a given character in a list by the corresponding character in a second list, or deletes the character if there is no corresponding character. For example, to replace all occurrences of 'a' with 'x', 'b' with 'y', 'c' with 'z', and to delete all occurrences of 'd', 'e', and 'f'; the characters to be replaced are entered as 'abcdef', and the replacement characters are entered as 'xyz'. Only characters and not white space characters, such as blank, tab, and escape, should be entered in the Character list. An example grammar sentence illustrating this is shown below:

### Sentence

**Modify Value** for the target part  
p\_string  
 as follows: **Return (str) (str) Remove/Convert characters from source/target field**  
**Using string target part: p\_string**  
**Changing character(s) abcdef To the corresponding character(s) xyz**  
 <end> <end> <end> <>

The Remove Character filter specifies the character(s) to remove. The Remove Character filter simply returns a string without the specified string of characters. An example grammar sentence illustrating this is shown below:

### Sentence

**Modify Value** for the target part  
p\_string  
 as follows: **Return (str) (str) Remove/Convert characters from source/target field**  
**Using string target part: p\_string**  
**Changing character(s) abcdef To the corresponding character(s)**  
**Removing the characters**  
 <end> <end> <end> <>

Additional examples of this filter are shown in the table below:

p_string	Changing characters	Replacement characters	Result	Usage
abcdefgh	abc	xyz	xyzdefgh	replacement - translate
	abcdef	xyz	xyzgh	replacement and removal of characters
abcdabefg	abc	xyz	xyzdxyefg	multiple translation
	abcdef	xyz	xyzxyg	multiple translation and removal

**Note:** when an integer source part is mapped to a string target part, the filter performs an integer to string conversion prior to replacement/removal of characters.

## Default Value Initialization

---

To apply a default value to an unmapped Target part a value may be entered in the Default Value column for the Target part. This functionality is only available to fields of type string, varstring, integer, or float with scale defined. The generated code does not check if the numeric default value exceeds the maximum value allowed for that type.

Additionally, to apply a default value to an unmapped Target part a Set Default Value Target part filter may be applied.

### Sentence

**Set Default Value to user defined value 1234 <end> <end> <>**

To apply a default value of ALL(some value) to an unmapped Target part a Set Default Value Target part filter may be applied as illustrated below. This filter can apply to string or numeric target parts, however, the generated code does not check if the numeric part default value exceeds the maximum value allowed for that type. To avoid this maximum value from being exceeded, string parts should be used for large numeric values.

User entered values can be in the form of :

character-constant = any printable ASCII character

numeric-constant = 0 through 9

### Sentence

**Set Default Value to user defined value ALL: 5 for the length of the target part<end>  
<end> <>**

## Numeric Format Support

---

Additional properties are provided for handling separate sign fields and numeric cleansing.

### Separate Sign Field

---

An unsigned numeric Source part may be processed as a signed part by identifying the separate part name that maintains the sign for the numeric part. Two conversion properties, `sign_negative_part` and `sign_negative_values`, are used together on the unsigned Source part to

specify this condition. You can denote the signed value with any printable character, such as -, (, \*, etc; or an array of values. For information on entering an array of values, refer to the “Retrieving a Schema” section of the Data System Library for C/FS Handbook. These properties are summarized in the following table:

Conversion Property	Applied to:	Possible values:
<i>sign_negative_part</i>	Source Part (unsigned numeric)	Name of part containing sign character
<i>sign_negative_values</i>	Source Part (unsigned numeric)	Any character, such as -, (, *, etc. OR, and array of values

### Placing Number In Requested Form

There are several functions available for placing a target part in the desired format.

### Space fill or Zero fill

There are cases when the output needs to be right justified with either leading spaces or leading zeroes. When the Target part is a string, by default it will be left justified and may need space filling if desired. Application of this code block on variable length parts may produce undesirable results as the part will be zero-filled or space-filled the length of the part, hence losing its variable length characteristic.

To use the space/zero fill option:

- Attach the code block `CP_cfs111_CleanseData.1` to the conversion on the Properties page in the code block section.
- Set the conversion property `cip_c_part_post_move_cb` on the Target part. Use the appropriate value depending on the results desired, either `zero-fill` or `space-fill`.
- If the source part is an integer and target is a string, apply the Modify Value filter to return the string equivalent of the integer part as illustrated by the grammar sentence below.

#### Sentence

#### Modify Value for the target part

`p_string`  
as follows: `Return (str) (str) the string equivalent of the integer part p_integer.u all types.src database.eti-installation.cfs <end> <end> <end> <>`

Negative number  
with parentheses

Create a Target-part filter to place parentheses around a negative number. An example grammar sentence illustrating this is shown below (integer Source part to string Target part):

**Sentence**

**Modify Value for the target part**

p\_string  
as follows: If (str) p\_integer.u\_all\_types.src\_database.eti-installation.cfs is less than an integer value 0 Concatenate (str) operand a string constant ( additional concatenate operand(s) operand the string equivalent of the integer part p\_integer.u\_all\_types.src\_database.eti-installation.cfs <end> additional concatenate operand(s) operand a string constant ) otherwise return Return (str) the string equivalent of the integer part p\_integer.u\_all\_types.src\_database.eti-installation.cfs <end> <end> <>

To enclose with parentheses *and* remove the negative sign you must use the Remove/Convert character filter as follows:

**Sentence**

**Modify Value for the target part**

p\_string  
as follows: If (str) p\_integer.u\_all\_types.src\_database.eti-installation.cfs is less than an integer value 0 Concatenate (str) operand a string constant ( additional concatenate operand(s) operand Remove/Convert characters from source/target field p\_integer.u\_all\_types.src\_database.eti-installation.cfs Changing character(s) – To the corresponding character(s) Removing the characters <end> additional concatenate operand(s) operand a string constant ) otherwise return Return (str) the string equivalent of the integer part p\_integer.u\_all\_types.src\_database.eti-installation.cfs <end> <end> <end> <>

The Remove/Convert filter will automatically handle the integer to string conversion.

Show an explicit  
sign

If the Source part mapped is an integer, a Target part filter is required to return the string equivalent of the integer. Explicit negative signs automatically appear; however, in order to have explicit positive signs, a Target-part filter must be applied to concatenate the sign to the positive integer (integer Source part to string Target part) as in the following example:

**Sentence**

**Modify Value for the target part**

p\_string  
as follows: If (str) p\_integer.u\_all\_types.src\_database.eti-installation.cfs is less than an integer value 0 return Return (str) the string equivalent of the integer part p\_integer.u\_all\_types.src\_database.eti-installation.cfs <end> otherwise return Concatenate (str) operand a string constant + additional concatenate operand(s) operand Return (str) the string equivalent of the integer part p\_integer.u\_all\_types.src\_database.eti-installation.cfs <end> <end> <end> <>

When the Source part mapped is a string, create a Target-part filter to perform this transformation. An example grammar sentence illustrating this is shown below (string Source part to string Target part):

### Sentence

#### Modify Value for the target part

p\_string

as follows: If (str) the result of applying a function returning an integer integer equivalent of p\_string.u all types.src databse.eti-installation.cfs <end> is less than an integer value 0 Concatenate (str) operand a string constant – additional concatenate operad(s) operand p\_string.u.all types.src database.eti-installation.cfs <end> otherwise return Concatenate (str) operand a string constant + additional concatenate operand(s) operand p\_string.u all types.src database.eti-installation.cfs <end> <end> <end> <>

### Left Justify

---

Typically transformation is not required for left justification when the mapped source part is a string or an integer and the mapped target part is type string. When the mapped Source part is a string and the incoming data is already left justified, it will remain on the target side as left justified. With an integer, the source part will need a filter to transform the data to type string which will then left justify the data. However, if the source string data is not left justified then a Target-part filter exists which will left justify. The filter will trim leading spaces.

### Sentence

#### Modify Value for the target part

p\_string

as follows: Return (str) (str) Left Justify p\_string.u all types.src database.eti-installation.cfs <end> <end> <end> <>

### Right Justify

---

A Target part filter exists to right justify a string to string mapping. The grammar sentence is similar to the left justification shown above -- Replace Left Justify with Right Justify.

To right justify an integer to string mapping, the Target part filter “Right Justified string equivalent of the integer part” will convert the integer to string and right justify the data in the target part by padding with leading spaces. Application of this filter on variable length parts may produce undesirable results as the part will be space-filled the length of the part, hence losing its variable length characteristic.

### Sentence

#### Modify Value for the target part

p\_string

as follows: Return (str) (str) Right Justified string equivalent of the integer part p\_integer.u all types.src database.eti-installation.cfs <end> <end> <end> <>

### Date Format Support

---

Source and Target part filters are provided for all date formatting. The input date format must be specified through a Source Part filter. All other date transformations will occur on the Target side.

### *Date Format Conversion and Validation*

---

To convert and validate a Source date, the Source-side Date/custom functions filter must be applied. This will allow use of the Target-side date transform filters by placing the ifile in ISO format (CCYYMMDD). The Source part DAS type must be defined as date or string.

The Source part filter path is:

- Select “Date/custom functions”
- Select “Perform Date Transformations”
- Select “Format input date to internal format”
- Choose one of the source parts
- Choose one of the input date formats
- Enter the cutoff year for deciding whether the 2 digit year is 1900 or 2000, for example, 50. This means anything greater than 50 will be 1900, anything less will be 2000.

When the source date part to convert to ISO is less than the required part length of eight (CCYYMMDD), such as 98/2/21, a source virtual part must be created. The Date transformation filter is applied to the virtual part selecting the date source part to be converted to ISO. Then the virtual part must be mapped to the target date part.

When the input date is invalid, applying the following three conversion properties set at the conversion level can specify a default date:

- default\_year is a four-digit century and year (CCYY)
- default\_month is a two-digit month (MM)
- default\_day is a two-digit day (DD)

CCYY, MM, and DD must be numeric values. If the CVARs are not set, the default date is 19000101.

---

### *Input Date Formats*

The input date format can have the month, day, and year in all permutations, i.e. month-day-year, year-day-month, Julian date-year, year-Julian date.... The following applies to the input date formats:

- If the month and day are represented as numbers, and one or both are a single digit number, the date is expected to contain separators.
- The year can be a two digit year or a four-digit year.
- The month can be represented in a three-character abbreviation or as the full name of the month and is case insensitive. If the month is in non-numeric format, the day can be a single or double-digit number. Separators are not required.
- Separators can have multiple characters.

The following shows the acceptable input date formats where:

mm m	=	equal 1 or 2-digit month
dd d	=	equal 1 or 2-digit day
yy yyyy	=	year with or without century
mon	=	3-character abbreviation or full name of month
jjj	=	julian day
-	=	separator (can be any separator character(s))

Month-Day-Year

mmddyy, mmddyyyy, mm-dd-yy, mm-dd-yyyy

mm-d-yy, mm-d-yyyy  
m-dd-yy, m-dd-yyyy  
m-d-yy, m-d-yyyy  
mondyy, mondyyyyy,

mondyy, mondyyyy  
mon-dd-yy, mon-dd-yyyy

mon-d-yy, mon-d-yyyy

Day-Month-Year

ddmmyy, ddmmyyyy, dd-mm-yy, dd-mm-yyyy

d-mm-yy, d-mm-yyyy  
dd-m-yy, dd-m-yyyy  
d-m-yy, d-m-yyyy  
ddmonyy, ddmonyyyyy

dmonyy, dmonyyyy  
dd-mon-yy, dd-mon-yyyy

d-mon-yy, d-mon-yyyy

Year-Month-Day

yyymmdd, yyyyymmdd, yy-mm-dd, yyyy-mm-dd

yy-mm-d, yyyy-mm-d

Month-Year-Day

mmyydd, mmyyyydd,  
mm-yy-dd,  
mm-yyyy-dd

mm-yy-d, mm-yyyy-d  
m-yy-dd, m-yyyy-dd  
m-yy-d, m-yyyy-d  
monyydd,  
monyyyydd

monyyd, monyyyyd  
mon-yy-dd,  
mon-yyyy-dd

mon-yy-d,  
mon-yyyy-d

Day-Year-Month

ddyymm, ddyyyyymm,  
dd-yy-mm  
dd-yyyy-mm

d-yy-mm, d-yyyy-mm  
dd-yy-m, dd-yyyy-m  
d-yy-m, d-yyyy-m  
ddyymon,  
ddyyyymon

dyymon, dyyyymon  
dd-yy-mon,  
dd-yyyy-mon

d-yy-mon,  
d-yyyy-mon

Year-Day-Month

yyddmm,  
yyyyddmm,  
yy-dd-mm,  
yyyy-dd-mm

yy-d-mm, yyyy-d-mm

yy-m-dd, yyyy-m-dd  
yy-m-d, yyyy-m-d  
yymondd, yyyymondd

yy-dd-m, yyyy-dd-m  
yy-d-m, yyyy-d-m  
yyddmon,  
yyyyddmon

yymond, yyyymond  
yy-mon-dd, yyyy-mon-dd

yydmon, yyyydmon  
yy-dd-mon,  
yyyy-dd-mon

yy-mon-d, yyyy-mon-d

yy-d-mon,  
yyyy-d-mon

Julian-Year  
jjjyy, jjjyyyy

Year-Julian  
yyjjj, yyyy,jjj

### Target-side virtual parts

---

Target-side virtual parts have been enhanced to allow the user to select the virtual part's type, length, and default value rather than defaulting to a part type "string". The CFS DAS object was changed to include an additional grammar attachment:

Name: *cfs\_virtual\_tp*  
Type: Create Target Part

*PDPR Reported Problems Corrected in Release 1.1.3*

The following problems reported to exist in Release 1.1.2 of the DSL for C/FS have been corrected in Release 1.1.3 of the DSL.

PDPR	Issue Description	Resolution Description	Object Type
7670	<i>Right Justify filters not working.</i>	<i>right-justify-define right-just-declare right-just-declare-args filter-define-variables utility-definitions utility-declarations  cfs_modval_str_t_p</i>	<i>Template      Grammar</i>
7671	<i>initialization of all numeric values unavailable</i>	<i>cfs_default_value_t_p fill-part-with-numeric-default-value</i>	<i>Grammar Template</i>
7673	<i>Retrieve schema doesn't allow for "add properties" function.</i>	<i>das/cfs.1/cfsRetrieveSchema.tcl testaddprop1 sample schema with .prop file</i>	<i>Other</i>
7721	<i>Virtual Part Null Indicators for Virtual parts not working correctly</i>	<i>string-dimensions target-break-part-name init-virtual-part-string init-virtual-part-varstring init-virtual-part-integer init-virtual-part-float init-virtual-part-date</i>	<i>Template</i>
7774	<i>Default day is not being set as the day but rather taking the value of default_month property.</i>	<i>common-supp-variables-decl</i>	<i>Template</i>
7819	<i>Numeric format support; separate sign isn't working.</i>	<i>handle-separate-sign-field define-handle-sep-sign-flt-cond cfs-move-ex-in-num-part cfs-move-external-internal-define multi-part-numeric-cleansing</i>	<i>Template</i>
7861	<i>Not all input date formats are being converted to ISO properly.</i>	<i>fcn-ParseGregDateAlpha-defn fcn-ParseJulianDate-defn fcn-format-indate-to-ISO-defn fcn-ValidateISOdate-defn</i>	<i>Template</i>
7908	<i>Null source date part transformed to ISO incorrectly.</i>	<i>fcn-format-indate-to-ISO-defn ex_format_indate_to_ISO-decl-args  modval_date_s_p</i>	<i>Template  Grammar</i>
7914	<i>Placing Number in Requested Form --- Negative number with parentheses not working and negative sign becomes embedded when applying zero-fill cip</i>	<i>cfs_modval_str_t_p CP_cfs111_CleanseData</i>	<i>Grammar Code Block</i>

7949	<i>Date formatting to ISO fails when source part is less than length of 8</i>	<i>cfs_modval_date_s_p</i>	<i>Grammar</i>
8023	<i>Target side virtual parts ability to select part type, length, default value, and annotation.</i>	<i>CFS</i>	<i>DAS Object</i>

**Note:** Object Types is intended to convey the area in which the problem resolution has been focused. It should also reflect the area to reference in this manual for more detail of the resolution.

*Additions and  
Changes to  
Template  
Functions and  
Grammars in  
Release 1.1.3*

The table below lists templates and grammars that have been added in Release 1.1.3.

Template Name	Template or Grammar affected	Description	P DPR
<i>cfs_virtual_t_p</i>	<i>grammar</i>	<i>Allow target side virtual part to specify part attributes.</i>	8023
<i>multi-part-numeric-cleansing</i>	<i>cfs_query</i>	<i>Check all parts for the presence of the property <i>sign_negative_part</i></i>	7819
<i>right-just-declare</i>	<i>cfs_populate</i>	<i>declare for right-justify function</i>	7670
<i>right-just-declare-args</i>	<i>cfs_populate</i>	<i>decides if using ansi standards for declare arguments</i>	7670
<i>right-justify-define</i>	<i>cfs_populate</i>	<i>the right justify function</i>	7670

The table below lists templates and grammars that have been changed in Release 1.1.3.

Template Name	Template or Grammar Affected	Description	P DPR
<i>cfs_default_value_t_p</i>	<i>grammar</i>	<i>Modified to allow a user defined default value of ALL character constant or numeric constant.</i>	7671
<i>cfs_modval_date_s_p</i>	<i>grammar</i>	<i>Modified to provide list of parts to select date source part to transform.</i>	7949
<i>cfs_modval_str_t_p</i>	<i>grammar</i>	<i>modified to allow right justification of integer to string conversion and modified to correctly concatenate '(' before and after a negative number and remove the '-' sign</i>	7670 7914
<i>cfs-move-ex-in-num-part</i>	<i>cfs_query</i>	<i>Corrected the placement of the call to TPL <i>handle-separate-sign-field</i> due to 2-part filter.</i>	7819
<i>cfs-move-external-internal-define</i>	<i>cfs_query</i>	<i>Call to new TPL <i>multi-part-numeric-cleansing</i> added.</i>	7819

<i>common-suppl-variables-decl</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Used the default_day property for the default day.</i>	7774
<i>CP_cfs111_CleanseData</i>	<i>CodeBlock</i>	<i>modified zero-fill to move sign to beginning of field before filling with leading zeros</i>	7914
<i>define-handle-sep-sign-flt-cond</i>	<i>cfs_query</i>	<i>Enhanced to allow the properties sign_negative_values as either a single value placed on the conversion or an array of values (either placed on the conversion or set via RS add props function) Changed to allow sign to be more than one byte, as in "CR"</i>	7819
<i>ex_format_indate_to_ISO-decl-args</i>	<i>cfs_query</i>	<i>Declared additional argument for the ex_format_indate_to_ISO function.</i>	7908
<i>fcn-format_indate-to-ISO-defn</i>	<i>cfs_query</i>	<i>Altered processing sequence to handle Julian dates greater than 5 or 7. Add additional argument to transform NULL date to default date.</i>	7861 7908
<i>fcn-ParseGregDateAlpha-defn</i>	<i>cfs_query</i>	<i>Corrected the following problems: Certain dates with alphabetic month returning invalid year and day. mdy/myd with alphabetic month returning invalid day.</i>	7861
<i>fcn-ParseJulianDate-defn</i>	<i>cfs_query</i>	<i>Corrected the following problems: Julian date not transformed correctly if date is greater than 5 or 7 (i.e: jjyy, jjyyyy). Invalid day returning month &amp; day from previous record.</i>	7861
<i>fcn-ValidateISOdate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Corrected DAY validation</i>	7861
<i>fill-part-with-numeric-default-value</i>	<i>cfs_populate</i>	<i>Modified to check for the existance of a sign for the part</i>	7671
<i>filter-define-variables</i>	<i>cfs_populate</i>	<i>added a definition of temporary modval buffer area</i>	7670
<i>handle-separate-sign-field</i>	<i>cfs_query</i>	<i>STOP_ITER was causing ITER to end prematurely</i>	7819
<i>init-virtual-part-date</i>	<i>cfs_query</i>	<i>Modified to either set the virtual part to a DSL based default value or accept the user specified value. Null indicator will no longer get set on.</i>	7721
<i>init-virtual-part-float</i>	<i>cfs_query</i>	<i>Modified to either set the virtual part to a DSL based default value or accept the user specified value. Null indicator will no longer get set on.</i>	7721
<i>init-virtual-part-integer</i>	<i>cfs_query</i>	<i>Modified to either set the virtual part to a DSL based default value or accept the user specified value. Null indicator will no longer get set on.</i>	7721

<i>init-virtual-part-string</i>	<i>cfs_query</i>	<i>Modified to either set the virtual part to a DSL based default value or accept the user specified value. Null indicator will no longer get set on.</i>	7721
<i>init-virtual-part-varstring</i>	<i>cfs_query</i>	<i>Modified to either set the virtual part to a DSL based default value or accept the user specified value. Null indicator will no longer get set on.</i>	7721
<i>modval_date_s_p</i>	<i>grammar</i>	<i>Added additional argument for the ex_format_indate_to_ISO function.</i>	7908
<i>string-dimensions</i>	<i>cfs_populate</i>	<i>Corrected target part name reference.</i>	7721
<i>target-break-part-name</i>	<i>cfs_populate</i>	<i>Corrected target part name reference.</i>	7721
<i>utility-declarations</i>	<i>cfs_populate</i>	<i>added reference for right-justify function</i>	7670
<i>utility-definitions</i>	<i>cfs_populate</i>	<i>added reference for right-justify function</i>	7670

## Handbook Updates for release 1.1.3

---

### Create a Schema Properties File (2/7)

The ETI\*EXTRACT Set Schema Properties Grammar reads an input file containing Schema Property values and creates Schema Definition Language (SDL) commands to set the specified Schema Property to the specified value for the specified database, unit, or part.

This section provides an overview of the syntax of the Set Schema Properties file with relevant examples. The Set Schema syntax supports setting Schema Properties to simple scalar values, arrays, and arrays of properties. The syntax does NOT include support for setting the value of a Schema Property to include a reference to a slot (such as <TPL x>).

#### Set Schema Properties Grammar Basic Syntax

Set Schema Properties Grammar Basic Syntax rules are shown below. They represent the syntax for setting individual properties and arrays of properties on the database, unit, and/or part.

#### Basic Syntax

db: **db name**

property: value&

unit: **unit name**

property: value&

part: **part name**

property: value&

The & marks the end of the property and must be present.

The db level is optional. If no property needs to be set at the database level, then the db level may be omitted. The unit level is required if setting a property on the unit or a part within the unit.

For a scalar property, value is the actual value to assign to the property.

For an array enclose the whole array in parentheses and separate each value with |, as shown:

```
cvar_name: (value1 | value2 | value3) &
```

To create an array of properties with each element of the array having either scalar or array properties, use the C\_VARS-ARRAY keyword:

```
C_VARS-ARRAY array_name
```

```
[ cvar_name1: value &
```

```
  cvar_name2: (value1 | value2 | value3) & ]
```

```
[ cvar_name1: value &
```

```
  cvar_name2: (value1 | value2 | value3) & ]
```

**Note:** The braces [ ] mark the beginning and end of each occurrence of the array.

To create an array of properties and scalar properties on the same schema object, list the array properties first using the C\_VARS-ARRAY keyword followed by the keyword C\_VARS:

```
C_VARS-ARRAY array_name
```

```
[ cvar_name1: value &
```

```
  cvar_name2: (value1 | value2 | value3) & ]
```

```
[ cvar_name1: value &
```

```
  cvar_name2: (value1 | value2 | value3) & ]
```

```
C_VARS
```

```
  cvar_name3: value &
```

```
  cvar_name4: (value1 | value2 | value3) &
```

Set Schema Properties  
Grammar Examples

Set Schema Properties Grammar Syntax Examples are shown below. They represent the setting of individual properties and arrays of properties on the database, unit, and/or part.

1. To set properties on a database, unit and a part:

```
db: VET
data_file_DSNAME: vet.dat&
unit: DOCTOR
merge_nomatch: 01&
merge_duplicates: 04&
part: RECORD_TYPE
unit_id_value: 01&
```

2. To set properties on a unit, and part, but not the database:

```
unit: DOCTOR
merge_nomatch: 01&
merge_duplicates: 04&
part: RECORD_TYPE
unit_id_value: 01&
```

3. To set a property to an array of values:

```
unit: DOCTOR
related_units: (PATIENT|APPOINTMENT)&
```

4. To create an array of properties:

```
unit: DOCTOR
CVARS-ARRAY indexes
[ index_name: PRIMARY&
index_parts: (AMA_ID|SSN)& ]
[index_name: SECONDARY&
index_parts: (LAST_NAME, FIRST_NAME)& ]
```

5. To create an array of properties and individual properties:

```
unit: DOCTOR
CVARS-ARRAY indexes
```

```
[ index_name: PRIMARY&
index_parts: (AMA_ID | SSN) & ]
[index_name: SECONDARY&
index_parts: (LAST_NAME, FIRST_NAME) & ]
CVARS
related_units: (PATIENT | APPOINTMENT) &
```

### Validating the New Schema (2-9)

```
C/File System Retrieve Schema version 3.0
This retrieve schema requires a C/File System database
description file.
Enter just the name if the file is in the MetaStore das
directory.
Or simply enter the full path.

C/File System db description file:  ENTER DATABASE
DESCRIPTION Filename

This retrieve schema accepts an optional file for adding
schema properties (CVARS) to the schema.
Enter just the name if the file is in the MetaStore das
directory.
Or simply enter the full path.
Press RETURN or ENTER if you do not have an additional
schema properties file.

additional schema properties (CVARS) file: ENTER SCHEMA
PROPERTIES Filename
```

Figure 2-4. Prompt for C/FS Database Description File and Schema Properties File

Enter the filename of the database description. As the prompt indicates, if you copied the DDF to the \$EXTRACT\_ROOTDIR/MetaStore/db\_name/das/cfs. major directory (or another major version directory), you can simply enter the name of the file. If you copied it to a subdirectory within the major version directory, you can enter the relative path from the cfs.major directory to the file. For example, you could enter subdirectory-name/filename. Otherwise, you must enter the full pathname for the file.

After you enter the name of the file, the Retrieve-Schema program processes the database description file, processes the Schema Properties file, and generates an export file, which is then automatically imported into the workset you are working in. When you are finished, you must next

validate the imported schema, and associate it with the appropriate database. The procedures for doing so are described in the following two sections.

---

## Enhancements in Release

### 1.1.2

This section provides a summary of the additional features and updates in Release 1.1.2 of the DSL for C/FS.

- ❑ Functionality has been added to allow Single Step capabilities. For more information, read “Single Step Processing” on page 21.
- ❑ Functionality has been added to allow comparisons of two *NULL* terminated strings that may contain whitespace. For more information, read “Variable Length String Comparisons” on page 21.
- ❑ Content-sensitive cleansing is handled through use of a custom C function. For more information, read “Content- Sensitive Cleansing/ Translation” on page 21
- ❑ Custom C functions can be created for either the Source or Target side and used either as an internal link or as an external link. For more information, read “Custom Function Calls” on page 22
- ❑ Target side *Modify Value* filters are available for source part translations to convert or remove character(s) and return a substring of source part(s). For more information, read “Special Character Handling” on page 24
- ❑ Additional properties are provided for handling separate sign fields and numeric cleansing. For more information, read “Numeric Format Support” on page 25
- ❑ There are several functions available for placing a string target part in the desired format. For more information, read “Placing Number in Requested Form” on page 26
- ❑ Source and Target part filters are provided for all date formatting. For more information, read “Date Format Support” on page 27
- ❑ When working with strings defined as dates, you can process dates using addition and subtraction algorithms, and you can modify the final output to a user-defined format. For more information, read “Date Arithmetic and Comparisons” on page 28
- ❑ Aggregation of source data is provided during the populate step prior to writing the data to the Target database allowing ranking to store the top or bottom *n* records within a break group if so desired. For more information, read “Aggregation” on page 29
- ❑ Notes describing updates to the Handbook for Data System Library for C/File System, Release 1.1. For more information, read “Handbook updates for Release 1.1.2” on page 38
- ❑ Product Design Problem Reports (PDPR’s) fixed in Release 1.1.2. For more information, read “Reported Problems Corrected in Release 1.1.2” on page 43
- ❑ Notes on additions, updates, and deletions of template objects in Release 1.1.2. For more information, read “Additions and Changes to Template functions in Release 1.1.2” on page 46

## Single Step Processing

---

Functionality to allow Single Step capabilities is available with C/FS to C/FS conversions that do not contain *merge*, *split*, or *sort* instructions. Additionally, both the *query* and *populate* conversions must execute on the same Unix host. This is because this functionality was implemented through the use of named pipes in Unix.

This Single Step functionality is accomplished by executing both the *query* and the *populate* instructions within the *query* shell program. Before the *query* and *populate* programs are executed, a Unix named pipe is created for the *ifile*. Then the *query* program is executed in the background and the *populate* program is started in the foreground. Unix ensures synchronization between the two programs for the named pipe buffer. Once complete the process deletes the named pipe. Since a named pipe is used, which is a synchronized buffer, a complete *ifile* is never generated.

To initiate Single Step functionality within a conversion, set the conversion property *single\_step\_requested* equal to *1* or *true* at the conversion level.

## Variable Length String Comparisons

---

Functionality to allow comparisons of two *NULL* terminated strings that may contain whitespace is available through a target side filter. A new function called *vl\_safe\_strcmp* has been written to trim whitespace from the right of a *NULL* terminated string and then compare the two lexicographically. This comparison can be demonstrated by the following example:

```
Compare - "TEST" to "TEST      "  
vl_safe_strcmp("TEST", "TEST      ");
```

These two strings will evaluate as equal. It should also be noted that this function is using the C library function *isalnum* to determine the whitespace characters from the alphanumeric characters and that punctuation will also be trimmed.

To utilize this functionality, apply a target side *Modify Value* filter grammar:

- Select *Modify Value*
- Choose *If* or *Select*
- Choose the string to compare
- Select "*is the same as (= ASCII representation with whitespace trim)*"
- Choose a comparison string
- Choose the return action

## Content-Sensitive Cleansing/ Translation

---

Content-sensitive cleansing is handled through use of a custom C function. Refer to the "CUSTOM FUNCTION CALLS" section for explanation of its use.

An example of this functionality is parsing of address: city, state, and zip code.

When implementing this on the Source side:

- Create three virtual parts for City, State, and Zip Code, providing default values for them. Without default values, the null indicator associated with them will be set.
- Create a Source part filter on the part containing the address to call a custom function (refer to “CUSTOM FUNCTION CALLS” section).
- Map the three virtual parts to the Target parts.

When implementing this on the Target side:

- Create a Target Virtual part. The Virtual part should appear after the real target parts; this is so that the values returned from the custom function call do not get overwritten.
- Map the source part containing the address to the Virtual part
- Apply a Target part filter to call the custom function (refer to “CUSTOM FUNCTION CALLS” section)

---

## Custom Function Calls

Custom C functions can be created for either the Source or Target side and used either as an internal link or as an external link.

---

## Internal Link

An internal link C Function is identified within ETI•EXTRACT<sup>®</sup> either by an actual value, a code block, or a template. To identify a C function as an internal link, a filter is applied identifying the Custom Code to insert.

The Source filter path is:

- Select *Insert Custom Code*
- Choose the source of the code
- Choose the code placement selection

The Target filter path is:

- Select *Modify Value*
- Select *Return*
- Select *A Call to a Custom C Function*
- Choose the input parameters: *Call by value, Call by reference, or Dereference variable* (contents)
- Choose the output parameters: *Call by value, Call by reference, or Dereference variable* (contents)
- Enter the function name

The CIPs `cip_c_declarations` and `cip_c_definition` may be used to declare and define the function.

## External Link

---

An external link is used when the C code is defined outside of ETI•EXTRACT<sup>®</sup>; its functionality may be used by other applications outside of ETI•EXTRACT<sup>®</sup>.

The Source filter path is:

- Select *Date/custom function*
- Select *Call Custom Functions*
- Choose the input parameters: *Call by value, Call by reference, or Dereference variable (contents)*
- Choose the output parameters: *Call by value, Call by reference, or Dereference variable (contents)*
- Enter the function name

The Target filter path is:

- Select *Modify Value*
- Select *Return*
- Select *A Call to a Custom C Function*
- Choose the input parameters: *Call by value, Call by reference, or Dereference variable (contents)*
- Choose the output parameters: *Call by value, Call by reference, or Dereference variable (contents)*
- Enter the function name

To identify a C function as an external link, you must specify the `link_src` conversion property either on the conversion level or on the Source or Target database level depending on where it is used. The value of `link_src` must be the full Unix path name of the file that contains the function(s).

In addition, you must declare the function within the ETI•EXTRACT<sup>®</sup> generated code through the use of the CIP `cip_c_declarations`. This CIP is used either as a code point or code block and its value will be the valid C declaration of the external function. For example:

```
char *my_return_function(int, int*, char**, double);
```

When applying the filter on the Target side, you need to map all of the inputs to the function from the Source side to the Target part (usually a virtual part) as the output of the function. If the function returns more than one value, the target parts that you reference must be global to the function. For example, an address correction routine that takes in city, state, address, company name, and zip code as mapped source parts may return one long string target part or a structure pointed to by a target virtual part.

## Special Character Handling

Target side *Modify Value* filters are available for source part translations to convert or remove character(s) and return a substring of source part(s).

### Remove/Convert Characters

A Target part *Modify Value* filter is applied to specify *removal/translation* of characters from the value of the Source field. This function applies to string extract types only.

The Convert Characters filter substitutes a given character in a list by the corresponding character in a second list, or deletes the character if there is no corresponding character. For example, to replace all occurrences of 'a' with 'x', 'b' with 'y', 'c' with 'z', and to delete all occurrences of 'd', 'e', and 'f'; the characters to be replaced are entered as 'abcdef', and the replacement characters are entered as 'xyz'. Only characters and not white space characters, such as blank, tab, and escape, should be entered in the Character list. An example grammar sentence illustrating this is shown below:

```

Modify Value for the target part
  p_string
  as follows: Return (str) (str) Remove/Convert characters from source/target field
Using string target part: p_string
  Changing character(s) abcdef To the corresponding character(s) xyz
  <end> <end> <end> <>
    
```

The *Remove Character* filter specifies the character(s) to remove. The *Remove Character* filter simply returns a string without the specified string of characters. An example grammar sentence illustrating this is shown below:

```

Modify Value for the target part
  p_string
  as follows: Return (str) (str) Remove/Convert characters from source/target field
Using string target part: p_string
  Changing character(s) abcdef To the corresponding character(s)
Removing the characters
  <end> <end> <end> <>
    
```

Additional examples of this filter are shown in the table below:

p_string	Changing characters	Replacement characters	Result	Usage
abcdefgh	abc	xyz	xyzdefgh	replacement – translate
	abcdef	xyz	xyzgh	replacement and removal of characters
abcdabefg	abc	xyz	xyzdxyefg	multiple translation

	<i>abcdef</i>	<i>xyz</i>	<i>xyzxyg</i>	<i>multiple translation and removal</i>
--	---------------	------------	---------------	---

### Substring Functions

A Target part *Modify Value* filter is applied to return only a portion of a string by using the *Substring* or the *Subsequence until End of Line* filter. For *Substring*, the beginning and ending indexes must be specified, while the *Subsequence* requires only the beginning index. An example grammar sentence illustrating the use of *Substring* is shown below:

**Modify Value for the target part**  
**p\_string**  
 as follows: **Return (str) (str) The substring of the string**  
**p\_string.u\_all\_types.src\_database.eti\_installation.cfs** starting at index (an integer) **2** up to and including (an integer) **5**  
 <end> <end> <end> <>

### Numeric Format Support

Additional properties are provided for handling separate sign fields and numeric cleansing.

### Numeric Data Cleansing

To perform numeric cleansing, three new conversion properties are provided. The conversion property *part\_numeric\_clean* must be set, either on the Source part(s) or on the Source unit. When set on the Source unit, numeric cleansing is performed on all numeric Source parts for that unit. The property, *part\_decimal\_point*, may be set if the decimal point in the Source part is not denoted by a period. When this property is not set, the decimal point is assumed to be a period. The property *part\_negative\_chars* may be set if the sign is denoted by a character other than -, (), CR, or DB. These properties are summarized in the following table:

Conversion Property	Applied to:	Possible values:
<i>part_numeric_clean</i>	Source Part or Source Unit	<i>true, false</i>
<i>part_decimal_point</i>	Source Part	Any character, such as a period, comma, etc. (optional)
<i>part_negative_chars</i>	Source Part	Any character, such as -, (, *, etc. (optional)

During the cleansing process the following is true:

- ❑ All spaces are removed.
- ❑ A number is considered negative if:
  - There is a leading or trailing sign.
  - The number is within parentheses ().
  - There is a leading or trailing literal “CR” or “DB” .
  - The presence of any of the character specified with the property `part_negative_chars`.
- ❑ A number is considered positive when all of the negative conditions are not met.
- ❑ No support for international four letter currency symbols is provided.
- ❑ The decimal point is determined by the property `part_decimal_point` if it exists; otherwise, a period is used.
- ❑ Any character which is not a digit or decimal point is dropped.

### Placing Number in Requested Form

There are several functions available for placing a string target part in the desired format. Each function containing a number is described in detail below.

### Space fill or Zero fill numeric

When the Source part is an integer and the Target is a string, by default the a character translation of the Target part with left-justification occurs. There are cases when the output needs to be right justified with either leading spaces or leading zeroes.

To use the space/zero fill option:

- ❑ Attach the code block `CP_cfs111_CleanseData.1` to the conversion on the Properties page in the code block section.
- ❑ Set the conversion property `cip_c_part_post_move_cb` on the Target part. Use the appropriate value depending on the results desired, either `zero-fill` or `space-fill`.

### Left Justify

Transformation is not required for left justification when the mapped Source part is a string or an integer. When the mapped Source part is a string, the incoming data is already left justified. With an integer, the incoming data is full justified. However, a Target-part filter does exist which allows left justification. When the mapped Source part is a string, the filter will trim leading spaces.

**Modify Value for the target part**  
`p_string`

as follows: `Return (str) (str) Left Justify p_string.u all types.src database.eti-installation.cfs <end> <end> <end> <>`

---

## *Date Format Support*

Source and Target part filters are provided for all date formatting. The input date format must be specified through a Source Part filter. All other date transformations will occur on the Target side.

---

## *Date Format Conversion and Validation*

To convert and validate a Source date, the Source-side *Date/custom functions* filter must be applied. This will allow use of the Target-side date transform filters by placing the ifile in ISO format (CCYYMMDD). The Source part DAS type must be defined as date or string.

The Source part filter path is:

- Select *Date/custom functions*
  - Select *Perform Date Transformations*
  - Select *Format input date to internal format*
  - Choose one of the input date formats
  - Enter the cutoff year for deciding whether the 2 digit year is 1900 or 2000, for example, 50. This means anything greater than 50 will be 1900, anything less will be 2000.
- 

## *Input Date Formats*

The input date format can have the month, day, and year in all permutations, i.e. month-day-year, year-day-month, Julian date-year, year-Julian date.... The following applies to the input date formats:

- If the month and day are represented as numbers, and one or both are a single digit number, the date is expected to contain separators.
  - The year can be a two digit year or a four-digit year.
  - The month can be represented in a three-character abbreviation or as the full name of the month and is case insensitive. If the month is in non-numeric format, the day can be a single or double-digit number. Separators are not required.
  - Separators can have multiple characters.
- 

## *Output Date Formats*

The output format is specified through a Target Part filter. The Target part filter path is:

- Select *Modify Value*
- Select *Return*
- Select *Format a Date output parameters*
- Choose one of the output formats (see below for rules)
- Choose to have a separator or not
- Choose to include the century or not
- Select *Input Date*
- Choose one of the input date options (ISO format)

The following is true for output date formats:

- ❑ The Month, Day, and Year can be in any order and optionally contain a multiple character separator.
- ❑ The Month and Day will always be a two-digit number.
- ❑ The Year may contain the century or may not.
- ❑ The Julian day is always a three-digit number.

## Date Arithmetic and Comparisons

When working with strings defined as dates, you can process dates using addition and subtraction algorithms, and you can modify the final output to a user-defined format.

### Comparison of two dates

When comparing whether one date is older or newer than another, apply the *Modify Value* filter to the Target part with the dates to compare mapped to the Target part. The comparison will be lexicographic if the dates are ISO compliant – “19980101” is less than “19980201”. An example grammar sentence illustrating this is shown below:

**Modify Value for the target part**

**Date\_out**

As follows: **If (str) the result of applying a function returning a date** Returning a date **Input Date select source part p\_date1.u multibase.src database.eti-installation.cfs <end> precedes ( < ASCII representation) (OLDER date)** Returning a date **Input Date select source part p\_date2.u multibase.src database.eti-installation.cfs <end> return** **Format a Date output parameters** Date formatted output Output Month & Day will be a 2 digit number, Year may be 2 or 4 digits depending on whether the century needs to be specified. Format ... **Month-Day-Year <end> Separator / <end> Include century in Year** Returning a date **Input Date select source part p\_date2.u multibase.src database.eti-installation.cfs <end> <end> otherwise return** **Return Format a Date output parameters** Date formatted output Output Month & Day will be a 2 digit number, Year may be 2 or 4 digits depending on whether the century needs to be specified. Format ... **Month-Day-Year <end> Separator / <end> Include century in Year** Returning a date **Input Date select source part p\_date1.u multibase.src database.eti-installation.cfs <end> <end> <end> <>**

### Add/Subtract days from date

To add or subtract days to a string date, apply the *Modify Value* filter to the Target part. An add days grammar sentence illustrating this is shown below:

**Modify Value for the target part**

**p\_date**

as follows: **Return (str) (str) Returning a date** By performing date arithmetic **Add days to a date** **Select date select source part p\_date.u\_all\_types.src database.eti-installation.cfs** Days being A value entered by User **10 <end> <end> <end> <>**

The *Format a Date output parameters* selection may be specified prior to the *By performing date arithmetic* selection to specify the output date format. The input can be the mapped Source, a system date, a run date, or a user-defined ISO date. This function allows for leap year and calendar month changes.

### *Difference between two dates*

---

Given two dates, you may need to determine the number of days between them for aging data purposes. Given two string date values, this filter returns an integer value. If you apply this filter to a Target string part, the integer value is written as an ASCII string value of the number. If you apply the filter to a Target integer part, the number of days remains an integer. Outcome is undetermined if the Target is a float or time data type.

An example grammar sentence using an integer Target part is shown below:

**Modify Value for the target part**  
**p\_integer**  
as follows: **Return (int) (int) Value from date manipulation Date Difference Select start date Choosing a date (str) target part p\_date Select end date Run Date run date**  
<end> <end> <end> <end> <>

An example grammar sentence using a string Target part is shown below:

**Modify Value for the target part**  
**p\_date**  
as follows: **Return (str) (str) Returning a date By performing date arithmetic The difference between 2 dates First date being (CCYYMMDD) select source part p\_date.u\_all\_types.src\_database.eti-installation.cfs Second date being (CCYYMMDD) Run Date run date**  
<end> <end> <end> <>

### *Use System Date as Default*

---

A Source or Target part filter can be applied to return the Current Date. Current Date will return the System Date. A grammar sentence illustrating this is shown below:

**Modify Value for the target part**  
**p\_date**  
as follows: **Return (str) (str) Returning a date input Date System Date (Current Date) system date**  
<end> <end> <end> <>

### *Use a Specified Value: Run Date, Current Date, Specified Date*

---

A Source or Target part filter can be applied to return either the current date (same as system date), Run Date, or a User-Defined date. See the grammar sentence above.

### *Aggregation*

---

Aggregation of source data is provided during the populate step prior to writing the data to the Target database allowing ranking to store the top or bottom *n* records within a break group if so desired. Aggregate (summary)

data is written to auxiliary databases designated for aggregation. In C the file is a random file accessed by a B-tree index maintained by the aggregation logic. Ranking data is written to a subunit of the unit that defines the summary data. The subunit is with an auxiliary schema.

Use of aggregation is divided among the following steps:

- Planning the conversion
- Mapping
- Setting up the auxiliary Target databases
- Setting up the context variables
- Dimension parts and break keys
- Using the filter

### Planning the Conversion

---

There are several issues to consider when creating a conversion which uses the *Aggregation* filter in addition to any normal plans for mapping data.

#### 1. Prepare *Aggregation* criteria

First, prepare the criteria for creating the dimension parts (which define the break keys), the fact parts (which hold the sum, count, etc.), and the ranking data.

The most important of these steps is the creation of the dimensions, which is covered in greater detail in the Dimension Parts and Break Keys section.

#### 2. Prepare schemas

Next, prepare the target schemas for use during *aggregation*.

The schemas used in the populate step must contain at least one target database for detail records and at least one auxiliary output database for summary records. Each target unit (detail record) may have one or more related auxiliary aggregation units (summary records). Each auxiliary aggregation unit contains summary records created from the detail records from a single target (detail) unit.

Using auxiliary units to write summary records will generally reduce the number of *ifiles* needed for the conversion. This is because each target unit has an associated *ifile*, while auxiliary units are not associated with an *ifile*. The *aggregation* code gets the correct data to the auxiliary unit's output buffer. Since summary records will often be created from the same or very similar detail data, associating the auxiliary aggregation units with the detail unit reduces the number of *ifiles* in the conversion.

In many cases, a data warehouse contains only summary records. In this case you need to define a dummy detail record containing all of the parts necessary for *aggregation*. The Conversion Specialist maps from the

sources to this unit. Once the record has been processed by *aggregation*, the record can be rejected, causing it not to be written.

### 3. Setting up the auxiliary target databases

Using an auxiliary database in a conversion requires extra preparation in addition to creating and retrieving a schema, creating a database, and adding the database object to the conversion. For the purposes of this description, only target databases for use with *Aggregation* are covered.

To be used as an auxiliary database, the database must first appear as a normal target-only database within the conversion. Then the user must add an entry for the auxiliary database in the *aux.ctf* file and apply the conversion property *auxiliary* (with value *POPULATE*) to the auxiliary database in the conversion.

The auxiliary file setup is described in more detail in the Auxiliary database section.

### 4. Applying the *Aggregation* filter

After setting up the auxiliary database, the *Aggregation* filter may be applied. In brief, the user performs the following actions:

- Apply the *Set Target Aggregation* filter to the appropriate target unit.
- Select the auxiliary target database to hold *summary/ranking* data. (If you do not have the option of selecting an auxiliary target, the auxiliary file has not been properly set up. Refer to the Auxiliary database preparation instructions.)
- Enter the dimension key pairings (*Aux Key / Target Key*). The filter will convert the list of dimensions into a list of break keys. Delete any unwanted break key combinations and accept the final break key list.
- Enter the fact part pairings (*Aux Fact part / Target Fact part*), along with the *fact* function. The following functions are available as *fact* functions: *SUM*, *COUNT*, *COUNT\_RECORDS*, *MIN*, *MAX*, *FIRST*, *LAST*, *Post Aggregation Key*.
- Enter (optional) rankings and rank fact part pairings.
- Run the conversion and check the auxiliary file output (and the target file output if required).

These steps are discussed in more detail in the section on applying the filter.

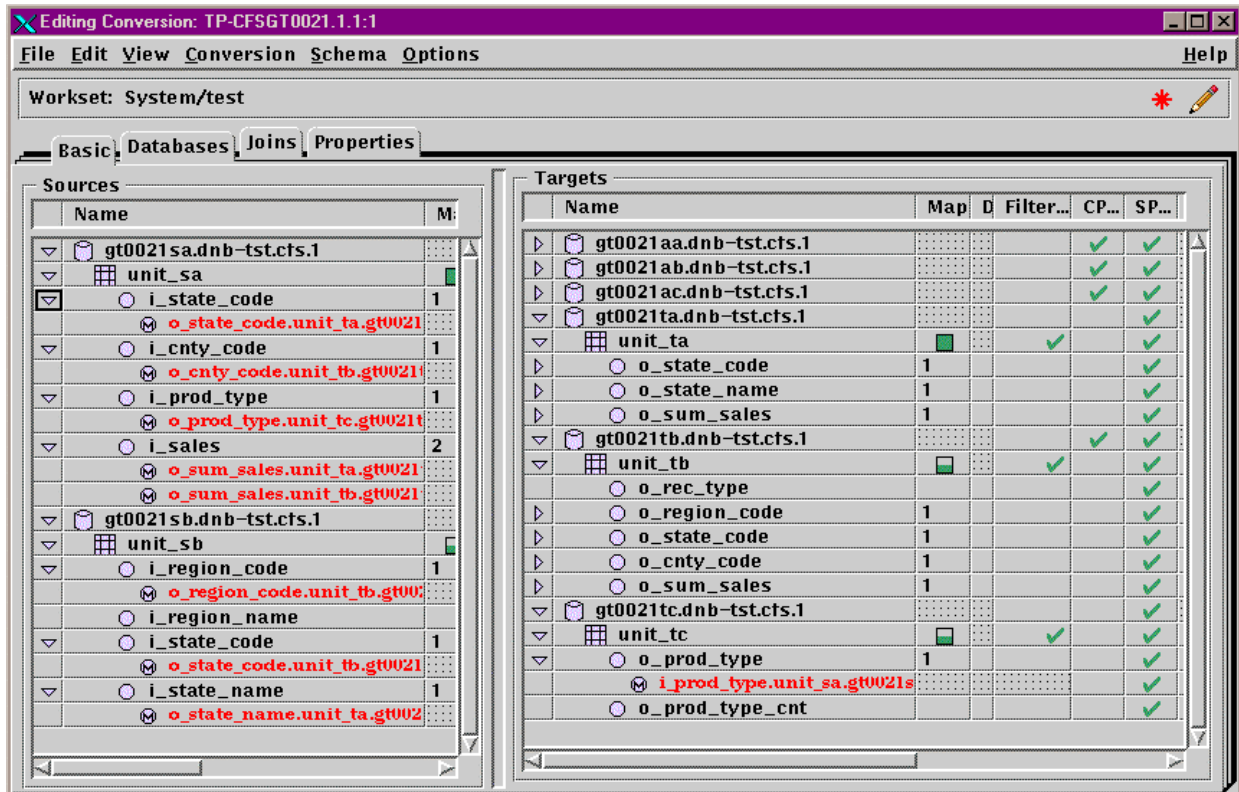
## Aggregation Mapping

---

Mapping for *Aggregation* to the target database is performed the same as all ETI•EXTRACT conversions. Map all relevant source parts to the target database. This is an explicit mapping, and all records are written to this database according to the standard conversion generated code. All break key parts, summation parts, and ranking parts are mapped to the target database. *NO parts are mapped to the auxiliary database.* The data

written to the target database populates the auxiliary target because of an *Aggregation* filter placed on a unit in the target database. The mapping to the auxiliary database is managed through implicit mappings from the target unit (not the source unit(s)) to the auxiliary target database.

More than one auxiliary aggregation unit can be linked to a target unit. The target unit will invoke each auxiliary aggregation to populate the auxiliary files.



### Setting Up the Auxiliary Target Databases

The following steps are required for defining the auxiliary database.

1. To be used as an auxiliary database, the database must first appear as a normal target only database within the conversion. Create a schema for the auxiliary database and retrieve the schema. Create a (normal) database in ETI•EXTRACT and set it to target only. Add this database to the conversion as a target database.
2. Add the context variable *auxiliary* to the auxiliary database. The only acceptable value for this context variable is *POPULATE*.
3. Configure the *aux.ct1* file. The file belongs in the *DAS* directory. (Look for *EXTRACT\_DIR/das/cfs.1/aux.ct1*). If this file does not exist in the *DAS* directory, copy the example *aux.ct1* file from the tape to the appropriate *DAS* directory. An example of an *aux.ct1* file is shown below:

```

# Format:
#
#
# "database-name" database.installation.das
#
# In the examples shown under SOURCE_PART and TARGET_PART, the
# following applies:
#
#     database-name:           = aux-summary
#     installation:            = c-tst
#     das:                     = cfs
#
# This file MUST be copied to the ../das/cfs.n/ directory.
#
# The database names you list here MUST be database objects in your
# MetaStore. Please refer to the CFS Handbook for how to construct
# your Auxiliary schemas that are to be attached to the database
# name.
#
# The database listed under source part below will be available
# to you upon filtering a source part. Likewise, the databases
# listed under target part below will be available to you upon
# filtering a target part.
#
#
# Do not change the line immediately following this line.
## SOURCE_PART
"aux-summary"                aux-summary.c-tst.cfs
#
#
# Do not change the line immediately following this line.
## TARGET_PART
"aux-summary"                aux-summary.c-tst.cfs

```

4. Edit the `aux.ct1` file. Add an entry under the `## TARGET_PART` line which includes the name of the auxiliary database in quotes and the auxiliary database in the form `database.installation.das`. Do not put the version number on the DAS. Be careful with the spelling of the information - check for dashes versus underscores, etc. The name and information in the `aux.ct1` file should match the Name entry for the database in the conversion. Do not delete the `##TARGET_PART` line, or any auxiliary database already defined

in the `aux.ct1` file.

There is only one `aux.ct1` file per DAS directory, and all auxiliary databases for the DAS are added to the same `aux.ct1` file. If the steps have been followed, you should see a list of the auxiliary databases available for the conversion when you run the *Aggregation* filter. If you are not given an option to select an auxiliary database and unit, review the previous directions. Pay close attention to spelling and capitalization when editing the `aux.ct1` file.

## Dimension Parts and Break Keys

---

The dimensions to be summarized into an auxiliary aggregation unit determine the parts used for break key processing. The user enters the dimensions and the *Aggregation* filter prompts with a list of the break keys which it can generate from the dimension part list.

Each dimension consists of one or more parts that make up a hierarchy. One or more dimensions may be entered in an *Aggregation* filter. For example, a locality dimension might consist of the parts STATE, COUNTY, and ZIP CODE. This is a valid dimension because states are made up of counties and counties have zip codes. In this example, we are stating that zip codes that cross county (or state) boundaries should be aggregated based on the county. For example, zip code 78729 is used in Travis and Williamson counties in Texas. Any detail record containing ZIP CODE 78729 will be aggregated into either the Travis or Williamson COUNTY break. There will not be a single summary record containing all the data for ZIP CODE 78729.

You can define break key processing for all of the keys in the dimension (the default), or you can remove certain parts from break key processing. For example: Given the locality dimension STATE, COUNTY, and ZIP CODE, the default action is to summarize by STATE; by STATE and COUNTY; and by STATE, COUNTY, and ZIP CODE. It may be that the STATE rollups are not required or that they can be calculated at query time. In this case you can include the STATE part in the dimension definition but remove it as a separate break level.

You may define multiple dimensions for a single auxiliary aggregation unit. Both single and multiple dimension breaks can be accumulated. By default, every possible break combination becomes a break key. For example:

Defining three dimensions as follows:

Dimension: STATE, COUNTY

Dimension: SIC2, SIC4, SIC6

Dimension: LOB, PRODUCT

results in rollups on 35 different breaks as follows:

STATE

STATE, COUNTY  
 SIC2  
 SIC2, SIC4  
 SIC2, SIC4, SIC6  
 LOB  
 LOB, PRODUCT  
 STATE, SIC2  
 STATE, SIC2, SIC4  
 STATE, SIC2, SIC4, SIC6  
 STATE, LOB  
 STATE, LOB, PRODUCT  
 SIC2, LOB  
 SIC2, LOB, PRODUCT  
 SIC2, SIC4, LOB  
 SIC2, SIC4, LOB, PRODUCT  
 SIC2, SIC4, SIC6, LOB  
 SIC2, SIC4, SIC6, LOB, PRODUCT  
 STATE, COUNTY, SIC2  
 STATE, COUNTY, SIC2, SIC4  
 STATE, COUNTY, SIC2, SIC4, SIC6  
 STATE, COUNTY, LOB  
 STATE, COUNTY, LOB, PRODUCT  
 STATE, SIC2, LOB  
 STATE, SIC2, LOB, PRODUCT  
 STATE, SIC2, SIC4, LOB  
 STATE, SIC2, SIC4, LOB, PRODUCT  
 STATE, SIC2, SIC4, SIC6, LOB  
 STATE, SIC2, SIC4, SIC6, LOB, PRODUCT  
 STATE, COUNTY, SIC2, LOB  
 STATE, COUNTY, SIC2, LOB, PRODUCT  
 STATE, COUNTY, SIC2, SIC4, LOB  
 STATE, COUNTY, SIC2, SIC4, LOB, PRODUCT  
 STATE, COUNTY, SIC2, SIC4, SIC6, LOB  
 STATE, COUNTY, SIC2, SIC4, SIC6, LOB, PRODUCT

The user has the ability to delete any of the break key definitions. In the preceding example, you could eliminate any of the 35 break keys.

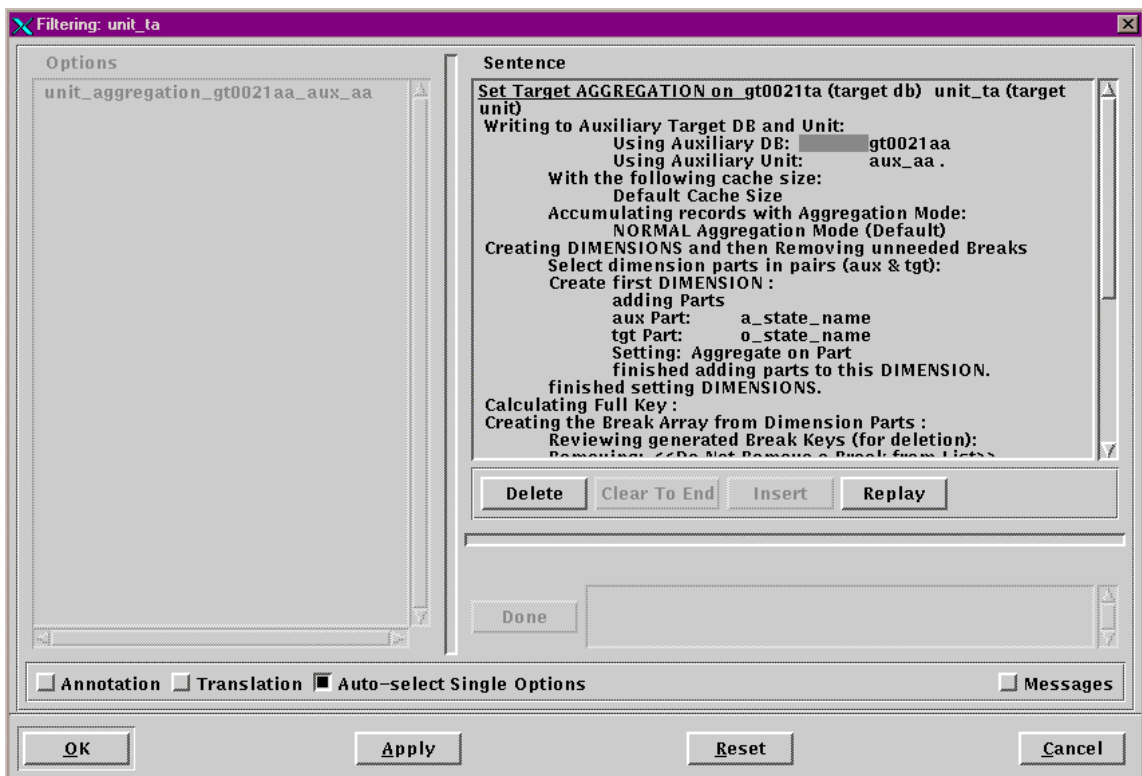
Since you can link multiple auxiliary aggregation units to a single target unit, it is possible to associate different break keys with different auxiliary

aggregation units. These units may be in the same or different auxiliary databases.

**Note:** You may enter dimension parts as *Aggregate* or *Do Not Aggregate*. Normally you enter parts with *Aggregate mode*. You must set the last part in a dimension to *Aggregate*, or else the process does not work properly. The *Do Not Aggregate* flag gives the Conversion Specialist a way to enter a part that is used as one piece of a multiple key dimension (or break key) but would not be used alone, or as the final key of a dimension or break key.

In the example above, setting STATE, SIC2, SIC4, and LOB to *Do Not Aggregate*, and COUNTY, SIC6, and PRODUCT to *Aggregate*, reduces the list to the following:

```
STATE, COUNTY
SIC2, SIC4, SIC6
LOB, PRODUCT
STATE, COUNTY, SIC2, SIC4, SIC6
STATE, COUNTY, LOB, PRODUCT
SIC2, SIC4, SIC6, LOB, PRODUCT
STATE, COUNTY, SIC2, SIC4, SIC6, LOB, PRODUCT
```



**Note:** It is never necessary to use the *Do Not Aggregate* flag. You can remove the other break key combinations in a later step of the *Aggregation* filter.

## Applying the Aggregation Filter

1. Apply the *Set Target Aggregation* filter to the appropriate target unit. Choose the auxiliary database which will receive the summary data. Do not apply the *Aggregation* filter to the auxiliary database itself.
2. Choose the auxiliary database and unit to populate with *summary/ranking* data. If you do not see any auxiliary databases or the appropriate auxiliary database, refer to the section on setup of the auxiliary database.
3. Enter the cache size (or accept the DEFAULT cache size). You can change this value for performance or space limitations.
4. Enter *NORMAL*, *REVERSE*, or *DATA\_DRIVEN Aggregation* mode. In *NORMAL* mode all of the fact functions behave normally. In *REVERSE* mode the *SUM* and *COUNT* functions subtract amounts or counts respectively, but the other fact functions are unaffected. In *DATA\_DRIVEN* mode the value of a data-driven field determines whether *NORMAL* or *REVERSE* mode is employed on a per-record basis. If you choose *DATA\_DRIVEN* you must enter the target part whose value governs *DATA\_DRIVEN* mode, and the value for that part which indicates *REVERSE* mode. Any other value indicates *NORMAL* mode.
5. Enter dimension parts. You must enter the dimension key pairings (*Aux Key part / Target Key part*) of dimension parts. The filter converts the list of dimensions into a list of break keys. Delete any unwanted break key combinations and accept the final break key list. See the section on Dimension Parts and Break Keys for a complete description. You may enter multiple parts per dimension and multiple dimensions per filter.
6. Remove unwanted Break Key combinations. From a list of Break Keys derived from the dimension parts, remove any unwanted combinations, or select *Do not Remove a Break from List* to accept the list.
7. Enter fact part pairings (*Aux Fact part / Target Fact part*), along with the fact function. The following functions are available as fact functions: *SUM*, *COUNT*, *COUNT\_RECORDS*, *MIN*, *MAX*, *FIRST*, *LAST*, and *Post\_Aggregation\_Key*. You may enter multiple fact part pairings, but do not duplicate the use of an auxiliary fact field (or data may be overwritten). Also, do not use the auxiliary key fields as fact fields as this will produce unpredictable break key results. The meanings of the functions *SUM*, *MIN*, *MAX*, *FIRST*, and *LAST* are apparent from their names. *COUNT* counts only records which contain data in the specified field, whereas *COUNT\_RECORD* counts all records regardless of whether the specified field contains data. The

*Post\_Aggregation\_Key* function does nothing as a function; it merely serves as a placeholder in the code for a value that can be modified later (for example, in a code block).

8. Enter *Rank* part pairings with associated *Value* part pairings and functions. You may add ranking values, but these are optional. You are prompted for the number of records to keep (top or bottom *n* records). Then enter a rank key part pairing (*Aux Rank key part / Target Rank key part*) and the associated *rank* value part pairings along with their *value* part fact functions. You may enter multiple *Value* parts per *Rank* part. You can only enter one *Rank* key part per ranking, but you may enter multiple rankings in any one *Aggregation* filter.

The *Aggregation* filter is now complete. You may apply more than one *Aggregation* filter to any one target unit, but the filters must select different auxiliary aggregation units to populate.

The auxiliary database concept has been extended to C/FS for use in *Aggregation*. However, C/FS does not support subunits, which are necessary to support the *Aggregation* function of ranking. In order to do *Aggregation* with ranking, use the following workaround. The auxiliary database is used in the *Aggregation* filter to gather information. It is also used as a schema template to create a different auxiliary output file and a new schema which implements ranking without subunits.

The *Aggregation* output in C/FS does not match the schema of the auxiliary database. A schema which matches the output data is created automatically, and is derived from the schema of the auxiliary database and the *Aggregation* filter itself.

## Handbook updates for Release 1.1.2

---

This section describes updates to the Handbook for Data System Library for C/File System, Release 1.1:

ETI AnswerLine™ information has been updated to reflect recent changes. For more information, read “Contacting ETI” on page 1

Items underlined in the following sections are corrections to the descriptions contained in the Handbook for the Data System Library for C/File System Release 1.1; all other items are additions.

### **Database Description File Format**

DDF Syntax (2-3)

*unit\_name* specifies the name of the unit as it will appear in the conversion editor. Be careful not to use any illegal characters in the *unit\_name*. For example, dashes are not allowed while the use of the underscore character is permitted.

*part\_name* specifies the name of the part as it will appear in the conversion editor. Be careful not to use any illegal characters in the *part\_name*. For example, dashes are not allowed while the use of the underscore character is permitted.

**Note:** Duplicate part names are not allowed. If a duplicate part name(s) exist, the properties from the second (and subsequent) instance(s) of the part will be applied to the initial instance of the part. Also, the unit’s record length will not be adjusted, which may cause undesirable results during the conversion.

Setting up Schemas for Multiple Record Types (2-7)

Positional hierarchy input structures require additional changes (as shown below) to the schema once the *DDF* has been retrieved. If the input structure is not related by position, no changes to the schemas are required. Target-side multiple record types are not currently supported in the *C/FS DSL*; each unit processed becomes a separate output file. A Code Insertion Point (CIP) is required that will recognize all record types. Refer to the *multi\_record\_processing* CIP in Appendix A.

Positional hierarchy schema changes:

- Within the Schema Editor, manually remove the Root designation from the child unit(s).
- Add a schema join from the *parent unit / key part* to the child unit (no part specified).

**Appendix A, Code Insertion Points and Context Variables**

*Program CIPs (A-11)*

Table A-17. Code Insertion Point: *c\_declarations*

<i>Program</i>	<i>query, populate</i>
----------------	------------------------

**Condition CIPS**

Code Insertion Point: *multi\_record\_processing (A-13)*

The following example will test whether the record just read from the *client* database file represents the unit *person* or *company* by checking the first character for *P* or *C*. Note that all *CVAR* values for *current\_db*, *current\_unit*, and all *current\_db.current\_unit* (record types) must be defined. The error “Error: unknown record type” is generated when a record type isn’t defined. The example shows the values for the CIPs followed by the code generated from it.

## CODE INSERTION POINTS

Code Insertion Point, *multi\_record\_processing*:

```
memset(type, (int) '\\0', 1);
<EVAL set CFS_UNIT_TYPE_client_person 1
><EVAL set CFS_UNIT_TYPE_client_company 2
> <EVAL set current_db client
>strncpy(type, <TPL cfs-internal-storage-glo-
bal-name>.buf_ptr + 0 ,1);
if(<TPL cfs-io-return-code-name> != EOF) {
    if ( strcmp(type,"P",1)==0 ) {
        <EVAL set current_unit person
><TPL das-unit-type-name> = <TPL das-curr-unit-type-indica-
tor>;
    } else {
        if ( strcmp(type,"C",1)==0 ) {
            <EVAL set current_unit company
><TPL das-unit-type-name> = <TPL das-curr-unit-type-indica-
tor>;
        } else {
            <TPL das-unit-type-name> = 0;
            append_warning("Error: Unknown record type\n");
            return 1;
        }
    }
}
```

Code Insertion Point: declarations:

```
char type[1];
```

## GENERATED CODE

```
memset(type, (int) '\\0', 1);
strncpy(type, src_mult_hier_cfs_globals.buf_ptr + 0 ,1);
if(src_mult_hier_cfs_globals.io_return_code != EOF) {
    if ( strcmp(type,"P",1)==0 ) {
        cfs_unit_type_src_mult_hier = 1;
    } else {
        if ( strcmp(type,"C",1)==0 ) {
            cfs_unit_type_src_mult_hier = 2;
        } else {
            cfs_unit_type_src_mult_hier = 0;
            append_warning("Error: Unknown record type\n");
            return 1;
        }
    }
}
```

### Context Variables (A-17)

Context Variable	Description	Usage/Additional Information
<i>auxiliary</i>	<i>This context variable is used to identify an auxiliary database. Its' only value allowed is "POPULATE".</i>	<i>Define as a property at the database level in the CE or EE.</i>
<i>data_file_DSNAME</i>	<i>The name of the data file to use for I/O. The default file names are <i>dbname.in</i> for Query programs, <i>dbname.unitname.out</i> for Populate programs.</i>	<i><u>For Query, define as a property at the database level in the CE to override the default file names. For Populate, define as a property at the database or unit level in the CE to override the default file names. For example, you can set the file name to payroll.dat.</u></i>
<i>default_day</i>	<i>This context variable is used when assigning a default date. It is a two-digit day number. Use in conjunction with <i>default_month</i> and <i>default_year</i>.</i>	<i>Define as a property in the CE or at the database level in the CE or EE.</i>
<i>default_month</i>	<i>This context variable is used when assigning a default date. It is a two-digit month number. Use in conjunction with <i>default_day</i> and <i>default_year</i>.</i>	<i>Define as a property in the CE or at the database level in the CE or EE.</i>
<i>default_year</i>	<i>This context variable is used when assigning a default date. It is a four-digit century and year. Use in conjunction with <i>default_day</i> and <i>default_month</i>.</i>	<i>Define as a property in the CE or at the database level in the CE or EE.</i>
<i>link_src</i>	<i>This context variable is used when defining external function calls. The link source defines the object code of the external function to call.</i>	<i>Define as a property at the source database level, or in the CE if it applies to calls on the source or target side; and, enter the entire Unix pathname.</i>
<i>part_decimal_point</i>	<i>This context variable is used to specify the printable character that denotes the decimal point. Use in conjunction with the <i>part_numeric_clean</i> property.</i>	<i>Define as a property at the source part level in the CE.</i>

<i>part_numeric_clean</i>	<i>This context variable is used to specify the precision of a source part string to be considered as a float variable. The value is either "true" or "false". Use in conjunction with the part_decimal_point property.</i>	<i>Define as a property at the source part level in the CE.</i>
<i>sign_negative_part</i>	<i>This context variable is used to specify that an unsigned source part is negative. The value is the name of the part. Use in conjunction with the sign_negative_values property.</i>	<i>Define as a property at the source part level in the CE.</i>
<i>sign_negative_values</i>	<i>This context variable is used to specify the printable character, such as -, (, *, etc., that denotes the sign. Use in conjunction with the sign_negative_part property.</i>	<i>Define as a property at the source part level in the CE.</i>
<i>single_step_requested</i>	<i>This context variable is used to request a conversion to be run in "single step" mode. (see enhancements descriptions.)</i>	<i>Define as property at the conversion level. Set 1 as true.</i>

*Appendix B, Messages (B-10)*

ETI-CFS-RE-8028      User Response: Make sure that any CIPs applied are valid and correct as necessary. Verify schemas for correctness. Regenerate the program. If the error persists after regenerating the program again, contact the ETI Answerline for assistance.

*Reported  
Problems  
Corrected in  
Release 1.1.2*

The following problems reported to exist in Release 1.1.1 of the DSL for C/FS have been corrected in Release 1.1.2 of the DSL.

<b>P DPR</b>	<b>Issue Description</b>	<b>Resolution Description</b>	<b>Object Types</b>
@@	<i>Enhancement: Increased date handling filters and data aggregation.</i>	<i>Refer to tables "Additions and Changes to Template Functions and Grammars".</i>	<i>Templates, Grammars, Grammar Extensions, Code Block Aux.ctl</i>
4703	<i>Subunits are not supported in populate.</i>	<i>Refer to C/FS Handbook &amp; C/FS Release Notes 1.1.2</i>	<i>Documentation</i>
4797	<i>Non-root indicator in ddf files.</i>	<i>Refer to section "Handbook updates for Release 1.1".</i>	<i>Documentation</i>
5433	<i>Record length is not updated when removing duplicate part names during the retrieve schema process.</i>	<i>Refer to section "Handbook updates for Release 1.1".</i>	<i>Documentation</i>
5833	<i>C/FS package object references template modules that are not members of any template libraries</i>	<i>Removed several orphaned template modules see "Additions and Changes to Template functions for Release 1.1. on page 24.</i>	<i>Template</i>
5850	<i>Handbook says data_file_dsname can be set on unit.</i>	<i>Refer to section "Handbook updates for Release 1.1".</i>	<i>Documentation</i>
5946	<i>Sample8.ddf provided with C/FS DSL schema source file generates a "No context variable named &lt;part_das_postfix&gt;" message.</i>	<i>Template cfs-move-in-ex-variable-sign changed.</i>	<i>Template</i>
6015	<i>Added Functionality to allow Single step conversions within the C/FS DSL.</i>	<i>Refer to section "Enhancements for release 1.1.2".</i>	<i>Template, Documentation</i>

P DPR	Issue Description	Resolution Description	Object Types
6062	Compile error in <code>append_warning</code> function on HP-UX.	Templates changed: <code>append-warning-define</code> ; <code>WARNING-FILE-SUPPORT</code> ; <code>append-warning-define</code> ; <code>include-defines</code>	Template
6237	"Exception was thrown" with Unix RC-139 when processing large source data files.	Templates changed: <code>subseq-define</code> , <code>fcn-format-ISO</code> , <code>fcn-Parse-Address-defn</code> .	Template
6250	Filter to check for spaces on <code>ifile</code> field not working as expected.	see pdpr 6312.	Template and Grammar Extension
6312	Target filter putting literal "SPACES" as default value in string. (also addressed the default value of "ZERO" ).	Templates changed: <code>filt-gen-spaces</code> , <code>move-part-default-to-out-string</code> . Grammar: <code>cfs_default_value_t.p</code> . Grammar Extension: <code>DSL-filter-fcns</code> and <code>CFS-filter-fcns</code> .	Templates, Grammar, Grammar Extension
6704	<code>cfs</code> populate <code>abends</code> in <code>hp</code> environment.	Template changed: <code>string-function-defn</code> .	Template
6674	<code>cfs-move-in-ex-fixed-sign</code> should return NULL for unsigned parts.	Templates changed: <code>cfs-move-in-ex-fixed-sign</code> <code>cfs-move-in-variable-sign</code> .	Template
6685	Shell script generates environment variable different than C code.	duplicate by and fixed under 7151 below.	Template
6946	The <code>sample1.ddf</code> needs to have a <code>part_postfix</code> on the final part in the unit.	Template <code>move-in-ex-variable-sign</code> changed.	Template
6954	Source multiple record type documentation is incomplete.	Refer to section "Handbook updates for Release 1.1".	Documentation

P DPR	Issue Description	Resolution Description	Object Types
6954	<i>Source multiple record types fail when no positional hierarchy exists</i>	<i>Templates process-data-define and process-data-many-types changed. Template case-default created.</i>	<i>Template</i>
6989	<i>Error appears in the C/FS documentation code example.</i>	<i>Refer to section "Handbook updates for Release 1.1".</i>	<i>Documentation</i>
7151	<i>Setenv for shell and getenv for c code generate different values for db name consequently program cannot find input file.</i>	<i>Changes made to environment-var-name-db-only and environment-var-name.</i>	<i>Template</i>
7291	<i>Added functionality to compare two strings of different length. This is accomplished through "Modify Value" filters.</i>	<i>Refer to section "Enhancements for release 1.1.2"</i>	<i>Template, Grammar, Documentation</i>
7697	<i>Support for K&amp;R standards.</i>	<i>Templates changes and additions to support K&amp;R standards.</i>	<i>Templates</i>

**Note:** Object Types is an addition to this listing and is intended to convey the area in which the problem resolution has been focused. It should also reflect the area to reference in this manual for more detail of the resolution.

@@ - Represents enhancements that do not have a corresponding PDPR numbers.

*Additions and Changes to Template functions in Release 1.1.2*

The table below lists templates that have been added in Release 1.1.2.

Template	Template or Grammar Affected	Description	P DPR
<i>AddDays-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Function prototypes changed to support K&amp;R and ANSI standards.</i>	<i>7697</i>
<i>case-default</i>	<i>cfs_query</i>	<i>The error message displayed when a source unit record type is invalid.</i>	<i>6954</i>
<i>CompareDates-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Function prototypes changed to support K&amp;R and ANSI standards.</i>	<i>7697</i>
<i>compile-link-go-ss</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	<i>This template allows iterating over instructions to do both query and populate (single step).</i>	<i>6015</i>
<i>compile-link-go-ss-pop</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	<i>This template will cause a message to appear in the shell stating that the entire conversion was handled in the query shell if single-step is true.</i>	<i>6015</i>
<i>compile-setup-ss</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	<i>This template handles creating the compile steps for both query and populate within the query shell for single-step.</i>	<i>6015</i>
<i>DifferenceOfDates-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Function prototypes changed to support K&amp;R and ANSI standards.</i>	<i>7697</i>
<i>display-wrn-file</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	<i>Added to handle display of wrn file.</i>	<i>6015</i>

Template	Template or Grammar Affected	Description	P DPR
<i>ex-alloc-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>execute-ss</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	This template handles setting up single step execution of both query and populate.	6015
<i>ex-format-into-date-to-ISO-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ex-format-ISO-to-outdate-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ex-free-to-mark-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ex-mark-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ex-num-data-cln-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ex-tr-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>find3LetterAbrevforMonth-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697

Template	Template or Grammar Affected	Description	P DPR
<i>findAlphaMonth-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>GenerateIntegerDate-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>GenerateInternalDate-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>GenerateInternalDateFromInteger-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>init_char_set_decl_args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>isNumeric-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>msg-file-name</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Generates message filename.	6015
<i>ParseGregDateAlpha-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ParseGregDateNumeric-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ParseJulianDate-decl-args</i>	<i>cfs_query</i>	Function prototypes changed to support K&R and ANSI standards.	7697

Template	Template or Grammar Affected	Description	P DPR
<i>set-enviro-vars</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Handles setting environment variables for query and populate for single-step.	6015
<i>set-single-step</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Iterates through instructions to set single-step flag.	6015
<i>set-ss-named-pipe</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Handles setting up making of named pipe.	6015
<i>strFindOffset-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>SubtractDays-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>sysdate-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>ValidateISOdate-decl-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Function prototypes changed to support K&R and ANSI standards.	7697
<i>validate-ss-instructions</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Checks all conditions necessary for a single-step to be possible.	6015
<i>vl-safe-strcmp-declare</i>	<i>cfs_query</i> <i>cfs_populate</i>	Call TPL <i>vl-safe-strcmp-declare-args</i> and <i>vl-safe-strcmp-name</i> to generate function prototype.	7291
<i>vl-safe-strcmp-declare-args</i>	<i>cfs_query</i> <i>cfs_populate</i>	Declares prototype under either K&R or ANSI C for function <i>vl-safe-strcmp</i> .	7291

Template	Template or Grammar Affected	Description	P DPR
<i>vl-safe-strcmp-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Contains base code for new function <i>vl-safe-strcmp</i> .	7291
<i>vl-safe-strcmp-name</i>	<i>cfs_query</i> <i>cfs_populate</i>	Returns name of new function <i>vl-safe-strcmp</i> to caller.	7291

**Note:** The template functions added for the date and aggregation enhancements were too numerous to list in the above table.

The table below lists templates that have been changed in Release 1.1.2.

Template	Template or Grammar Affected	Description	PDPR
<i>append-warning-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>append-warning-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified <i>strstr</i> function to eliminate compile errors on HP-UX.	6062
<i>cfs_default_value_t_p</i>	Grammar	Modified to put actual blanks (" ") in string default and 0 in numeric default rather than "SPACES" and ZEROES. Also added link to new Grammar Extension: <i>CFS-filter-fcns</i> in which an override was placed for validation routine: <i>fmt_str_tgt_part</i> .	6312
<i>cfs_modval_t_p</i>	Grammar	Modified to allow new filter options for <i>vl-safe-strcmp</i> .	7291
<i>cfs-external-out-buffer-define</i>	<i>cfs_populate</i>	Create global buffer area only once.	7697
<i>cfs-includes-defines</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>cfs-move-ex-in-num-part</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>cfs-move-external-internal-define</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>cfs-move-in-ex-fixed-sign</i>	<i>cfs_query</i> <i>cfs_populate</i>	Added EVAL <i>ifexists</i> slot on <i>part_das_postfix</i> .	6647 6674

Template	Template or Grammar Affected	Description	P DPR
<i>cfs-move-in-ex-variable-sign</i>	<i>cfs_query</i> <i>cfs_populate</i>	Added EVAL ifexists slot on part_das_postfix.	5946 6946 6674
<i>comment-populate-ifiles</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>common-date-functions-decl</i>	<i>cfs_query</i> <i>cfs_populate</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>compile-link-go</i>	<i>cfs-populate-shell</i>	Modified for new date and aggregation functionality.	@@
<i>compile-link-run</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Added check for single_step_requested and branch to related checks to see if single step is possible for this conversion.	6015
<i>declarations</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>display-files</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Added display of wrn file.	6015
<i>display-msg-file</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Changed from displaying wrn file to displaying msg file.	6015
<i>environment-var-name</i>	<i>cfs_query</i> <i>cfs_populate</i> <i>cfs_query_shell</i> <i>cfs_populate_shell</i>	Added prefix of 'ev' to unit name and database name in EVAL unique statement.	7151
<i>environment-var-name-db-only</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i> <i>cfs_query</i> <i>cfs_populate</i>	Added prefix of 'ev' to database name in EVAL unique statement.	7151

Template	Template or Grammar Affected	Description	P DPR
<i>fcn-format-ISO</i>	<i>cfs_query</i> <i>cfs_populate</i>	Added <i>strdup</i> call upon return from <i>subseq</i> to capture results of date substring.	6237
<i>fcn-Parse-Address-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Sample template function changed to handle <i>subseq</i> results correctly.	6237
<i>fcn-AddDays-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-CompareDates-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-DifferenceOfDates-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-find3LetterAbrevforMonth-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-findAlphaMonth-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-format-invalidate-to-ISO-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-format-ISO-to-outdate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697

Template	Template or Grammar Affected	Description	PDPR
<i>fcn-GenerateIntegerDate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-GenerateInternalDate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-GenerateInternalDateFromInteger-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-ifNotLeapYear-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-num-data-cln-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-ParseGregDateAlpha-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-ParseGregDateNumeric-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-ParseJulianDate-defn</i>	<i>cfs_query</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-SubtractDays-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697

Template	Template or Grammar Affected	Description	P DPR
<i>fcn-sysdate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>fcn-ValidateISOdate-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>filter-call</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>filter-declarations</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>filter-define</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>filter-define-number</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>filter-define-string</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>filter-define-variables</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>filter-definition</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>filt-gen-spaces</i>	<i>cfs_populate</i>	Changed iter over ifile parts to iter over all ifile parts not just first part. Also added iter over target parts.	6312
<i>include-defines</i>	<i>cfs_query</i> <i>cfs_populate</i>	Additional header files included: time.h & math.h.	6062

Template	Template or Grammar Affected	Description	P DPR
<i>init-program-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>init-virtual-part</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>internal-out-buffer-part-char-name</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>internal-out-buffer-part-name</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>int-in-buff-part-name-var-from-ifile</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>memory-functions-decl</i>	<i>cfs_query</i> <i>cfs_populate</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>memory-functions-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>misc-functions-decl</i>	<i>cfs_query</i> <i>cfs_populate</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>misc-functions-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>move-part-default-to-out-numeric</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@

Template	Template or Grammar Affected	Description	P DPR
<i>move-part-default-to-out-string</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality. Replaced actual " with <CVAR quotes> before & after CVAR pop_default_value.	@@ 6312
<i>move-part-in-to-out</i>	<i>cfs_query</i>	Modified for new date and aggregation functionality.	@@
<i>move-part-in-to-out-filtered</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>numeric-data-cls-function-decl</i>	<i>cfs_query</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>pop-function-free-mem</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>populate-date-functions-decl</i>	<i>cfs_populate</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>process-data-define</i>	<i>cfs_query</i>	Removed the error processing and placed in new template function "case-default". This problem relates to processing source-side multiple record types. Refer to the section "Handbook updates for Release 1.1" for details.	6954

Template	Template or Grammar Affected	Description	P DPR
<i>process-data-many-types</i>	<i>cfs_query</i>	Template function "case default" is called when an invalid record type encountered.	6954
<i>process-unit-define</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>query-date-functions-decl</i>	<i>cfs_query</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>storage-db</i>	<i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>string-translation-functions-decl</i>	<i>cfs_query</i> <i>cfs_populate</i>	Call a template function to set the function prototype based on ANSI or K&R standards.	7697
<i>string-translation-functions-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Changed the function definition to support ANSI and K&R standards.	7697
<i>string-function-defn</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified the <i>ex_tr</i> function to eliminate possible memory leaks.	6704
<i>subseq-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Improved the memory handling for the return string.	6237
<i>term-program-define</i>	<i>cfs_query</i> <i>cfs_populate</i>	Modified for new date and aggregation functionality.	@@
<i>utility-declarations</i>	<i>cfs_query</i> <i>cfs_populate</i>	Added TPL call to <i>vl-safe-strcmp-declare</i> .	7291

Template	Template or Grammar Affected	Description	PDPR
<i>utility-definitions</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Added TPL call to vl-safe-strcmp-define.</i>	7291
<i>utility-functions</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Modified for new date and aggregation functionality.</i>	@@
<i>warning-file-name</i>	<i>cfs_query_shell</i> <i>cfs_populate_shell</i>	<i>Changed to generate wrn filename instead of msg filename.</i>	6015
<i>warning-file-name</i>	<i>cfs-populate-shell</i>	<i>Modified for new date and aggregation functionality.</i>	@@
<i>WARNING-FILE-SUPPORT</i>	<i>cfs_query</i> <i>cfs_populate</i>	<i>Modified strstr function to eliminate compile errors on HP-UX.</i>	6062

The table below lists templates that have been removed in Release 1.1.2.

PDPR	Template
5833	functions.c_utilities base.cfs_c_cips base.cfs_c_common base.cfs_c_das_specific base.cfs_c_das_specific_message base.cfs_c_error_file base.cfs_c_exception base.cfs_c_heap base.cfs_c_messages base.cfs_c_populate base.cfs_c_populate_entry base.cfs_c_populate_patch base.cfs_c_string base.cfs_c_util_fcns

### Summary of Changes in Release 1.1.1

The following list summarizes the significant changes that have been implemented in Release 1.1.1 of the Data System Library for C/File System:

- ❑ A problem has been corrected in which the Populate program and shell script for running the Populate program referenced different environment variables.
- ❑ The Data System Library Handbook for C/File System contains inadequate information regarding processing of multiple record types. Additional information is provided below to supplement that found on page 2-7 of the Handbook. This information will be included in the next revision of the Handbook.

When retrieving a schema from a Data Definition File (DDF), the DDF should contain two or more separate unit definitions. In the following example, three record types are specified, each having a record key and a data part.

```

unit_A ... {
    record_type_key_part ... ;
    data_part ... ;
}
unit_B ... {
    record_type_key_part ... ;
    data_part ... ;
}
unit_C ... {
    record_type_key_part ... ;
    data_part ... ;
}
    
```

See Chapter 2 of the Handbook for additional information about setting up DDFs for schema retrieval. When the DDF is ready, do the following:

1. Retrieve the schema as usual. When the schema has been retrieved, all units are regarded as "root" units. All of the units except one must have this schema property removed. Selecting the root unit will depend on which unit should be the driver unit after a join has been applied (as explained in step 2). The example shown above for instance, would have the root specification removed from units B and C, but preserved in unit A.

To remove the root unit specification, select the multi-record schema from the list of available schemas and start the Schema Editor. In the "DAS Type" column beside the selected unit, double-click on the word "root", then select "Not Root" from the popup window. If the box is empty, then the unit has already been marked as a non-root unit. Repeat this step for all the other non-root units.

2. Set up the schema joins between the key part in the root unit and the other units. In the example above, the `record_type_key_part` in `unit_A` is joined with units B and C.
3. The handling of the multi-record types in the program code is still controlled through the use of CIPs, as defined by using the "multi\_record\_processing" CIP outlined in Appendix A of the Handbook for C/FS. See also the note in these Release Notes on connections to the CIP examples.
  - ❑ A problem has been corrected in which a null indicator was not reset. The affected template has been modified to include a reset of the null indicator for each part mapped in the conversion program.
  - ❑ Templates for Query and Populate programs were corrected to include checks for the existence and value of the Conversion Property **ansi\_c** in order to generate code specific to compiling with K&R C compilers. The Handbook for C/FS incorrectly states that the value of this Property must be **FALSE**. Instead, the value must be **0** (zero).
  - ❑ A problem has been found in which the Retrieve-Schema program fails if it encounters a `control-z` (displayed as `^z`) at the end of the DDF. This problem stems from creating DDFs on a personal computer and uploading them to your ETI•EXTRACT host. Some file transfer mechanisms append a `control-z` at the end of the file. You must use an editor to remove the `^z` characters before retrieving the schema.
  - ❑ A problem has been found in which angle-brackets (< and >) are not handled properly. The Retrieve-Schema program handles them correctly, but some ETI•EXTRACT tools may not. To work around this problem you should substitute some other characters for the angle brackets in the DDF and then edit the schema after it has been retrieved.

- ❑ A problem has been corrected in which the Retrieve-Schema program did not handle some pre- and postfix characters correctly. The program has been amended to handle additional pre- and postfix notations.
- ❑ The Retrieve-Schema program has been modified so that it now cleans a prefix slot by removing characters found after the closing quotation mark in the prefix specification.
- ❑ A problem has been corrected in which the Populate program would open all output files at the same time, resulting in the last file opened receiving all of the output data for the conversion. The populate templates have been corrected to open only the output files to which data is being written (in a specific order). Each output file opened is closed before the next output file is processed.
- ❑ If a conversion shows a loss in precision while handling DECIMAL numeric types, this may result from an incorrect schema representation. The precision specified in the DDF for a part may be too small. See Chapter 2 of the Handbook for C/FILE SYSTEM for a more detailed discussion of precision.

---

*Known  
Limitations of  
the DSL for  
C/File System*

This section outlines known limitations and problems of concern to users of the ETI•EXTRACT Data System Library for C/File System, Release 1.1.

---

*Conversion  
Annotation Field*

The annotation field for the conversion should not include /\* \*/ since this field is included by the ETI-provided shell scripts into the header.

---

*Input Data File*

Records that have a `unit_postfix` value of “\n”, that is, newline (or any other value) must have that same value at the end of the last record.

---

*Reading Input  
Data*

The DSL for C/File System relies upon the C functions `atoi` (alphabetic to integer) and `atof` (alphabetic to float) to read input data. These routines do not provide any warning or error messages if invalid input is passed to them. Consequently, no warning or error messages are provided by the DSL for C/File System when this situation arises. This limitation will be addressed in a later release of the DSL for C/File System.

For example, if the DSL for C/File System encounters the invalid data `26aaa` for an integer field, the following occurs:

- ❑ 26 is returned.
- ❑ No warning or error message is generated.
- ❑ The processing continues.

Because of the use of the C functions `atoi` and `atof`, there is a potential loss of precision when reading large numbers. This is especially true for floats if the value of the number being read is greater than the constant `FLT_MAX` (defined in the header `<float.h>` for your system).

---

*Differences  
Between Source  
and Target  
Schemas*

Multiple units and subunits are supported in source databases, but **not** in target databases.

---

*Multiple Record  
Type Processing*

The C/FS Query templates iterate over source units, which include only mapped units. Because of this, all units in a multiple record type must be mapped.

---

*Decimals*

Due to the possible loss of precision, decimals are not supported by the DSL for C/File System. There is, however, a decimal package available for users who wish to accept the risk.

Additionally, floats, which are treated internally as double, can be forced to print with decimal or scientific notation in the output files from the populate program. (See context variable `part_das_format_float` in Appendix A.)

---

*C Intermediate  
Actions*

The sort program does not sort floats (with exponential format) properly.

---

*Writing Integer  
Data*

C/FS reserves one character for the sign (a blank for unsigned data). Therefore, the minimum length on an integer target part is 2. To write a numeric field of length 1, define your target as a string of length 1 and apply a filter.