CASH\_HDR System

Version 1.0

System Documentation

01/20/2010

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Description | Author |
| 01/20/2010 | 1.0 | Original Document | Paul M. Kadow |

Contents

[1 Overview 6](#_Toc251827348)

[1.1 Purpose 6](#_Toc251827349)

[1.2 CASH Systems Concepts 6](#_Toc251827350)

[1.3 Description 6](#_Toc251827351)

[1.4 Data Elements 6](#_Toc251827352)

[1.5 CASH\_HDR System and HIDU System 7](#_Toc251827353)

[1.6 Terms 7](#_Toc251827354)

[1.7 Tables/Files/Globals Used 8](#_Toc251827355)

[2 Production Processes 9](#_Toc251827356)

[2.1 CASH\_HDR consists of 3 main processes 9](#_Toc251827357)

[2.2 Differences between ALL^CASH, START^CASH and CREATE^CASH 9](#_Toc251827358)

[2.3 Map all Vista Files – VISTA Method 10](#_Toc251827359)

[2.3.1 Overview 10](#_Toc251827360)

[2.3.2 (CASH.HDR.Utility).VISTA() – Initial call Overview 10](#_Toc251827361)

[2.3.3 (CASH.Utility).All – Map all Vista Files 10](#_Toc251827362)

[2.3.4 ALL^CASH 10](#_Toc251827363)

[2.3.5 ALL^CASHC 10](#_Toc251827364)

[2.3.6 ALL^CASH - (CASH.HDR.Utility).VISTA() - Description by Graphic Representation 10](#_Toc251827365)

[2.3.7 ALL^CASH - (CASH.HDR.Utility).VISTA() Description 12](#_Toc251827366)

[3 Technical Documentation 13](#_Toc251827367)

[3.1 Three primary elements of ALL^CASH - (CASH.HDR.Utility).VISTA() 13](#_Toc251827368)

[3.2 Key to understand ALL^CASH - (CASH.HDR.Utility).VISTA() 13](#_Toc251827369)

[3.3 Detailed Description of ALL^CASH - (CASH.HDR.Utility).VISTA() 13](#_Toc251827370)

[3.3.1 First reads from ^DIC/^DD AND Populate ^CASH(15050.11) 13](#_Toc251827371)

[4 Globals/Tables/Files 20](#_Toc251827372)

[4.1 ^CASH (“HDR”) Global 20](#_Toc251827373)

[4.2 CASH FM CLASS MAP Global ^CASH (15050.11) 20](#_Toc251827374)

[4.3 CASH CUSTOM DATATYPES Global ^CASH (15050.12) 20](#_Toc251827375)

[4.4 CASH ERRORS Global ^CASH (15050.13) 20](#_Toc251827376)

[4.5 CASH PARAMETER FILE Global ^CASH (15050.14) 20](#_Toc251827377)

[4.6 CASH LOCALIZATION FILE Global ^CASH (15050.17) 20](#_Toc251827378)

[4.7 CASH SQL RESERVED WORDS Global ^CASH (15050.19) 21](#_Toc251827379)

[4.8 CASH FM CLASS MAP Global ^CASH (15050.11) 21](#_Toc251827380)

[***4.8.1*** ***CASH FM CLASS MAP Global ^CASH (15050.11) - Detail*** 21](#_Toc251827381)

[***5*** ***FLAGS*** 26](#_Toc251827382)

[***5.1*** ***Compile - “C”*** 26](#_Toc251827383)

[***5.2*** ***Description - "D"*** 26](#_Toc251827384)

[***5.3*** ***Expand - “E”*** 27](#_Toc251827385)

[***5.4*** ***Force - “F”*** 27](#_Toc251827386)

[***5.5*** ***Override Names - “h”*** 27](#_Toc251827387)

[***5.6*** ***Simple IDs - “I”*** 27](#_Toc251827388)

[***5.7*** ***Loose Validation - “L”*** 27](#_Toc251827389)

[***5.8*** ***Multiples - “M”*** 27](#_Toc251827390)

[***5.9*** ***Simple Names - “N”*** 27](#_Toc251827391)

[***5.10*** ***Pointers - “P”*** 27](#_Toc251827392)

[***5.11*** ***SQL Only Compile - “Q”*** 27](#_Toc251827393)

[***5.12*** ***Recursive - “R”*** 28](#_Toc251827394)

[***5.13*** ***Partially Recursive - “r”*** 28](#_Toc251827395)

[***5.14*** ***SOAP Web Services - “S”*** 28](#_Toc251827396)

[***5.15*** ***Verbose - “V”*** 28](#_Toc251827397)

[***5.16*** ***Web Page - “W”*** 28](#_Toc251827398)

[***5.17*** ***XML - “X”*** 28](#_Toc251827399)

[***5.18*** ***PACKAGE (Optional)*** 29](#_Toc251827400)

[***5.19*** ***“ID” - (Optional)*** 29](#_Toc251827401)

[***5.20*** ***OWNER (Optional)*** 29](#_Toc251827402)

[***5.21*** ***LIST (Optional)*** 29](#_Toc251827403)

# Overview

## Purpose

The purpose of the CASH\_HDR system is to allow Caché SQL Queries easy access to the VistA data.



## CASH Systems Concepts

CASH System concepts:

1. CASH System - Native/Original Product created without regard to HDR processing. The VA developed the native CASH utility for creating M FileMan SQL Class Definitions. It was modeled after the original FileMan to Caché (FM2Caché ) tool by InterSystems.
2. CASH\_HDR System – original CASH System with HIDU extensions for HDR processing.
3. HIDU (as related to CASH): more customization added for updating the HDR database classes at the *VistA Database Sites*. (The Java application is a big part of HIDU but isn’t directly related to CASH\_HDR; the Java application was created to actually update the HDR VistA Database Sites)

This document refers to the CASH\_HDR System

## Description

*The CASH\_HDR System* creates Caché SQL Classes based on *VistA Data Dictionary Files (^DD, ^DIC)* that allows the SQL Classes to read Vista data in its M/native format.

## Data Elements

Caché SQL Classes/Tables will be created to emulate VistA Data Dictionary elements, specifically:

|  |  |  |
| --- | --- | --- |
| Converted From | Will be created to emulate: | Converted To |
| VistA File Definitions | Caché Classes |
| VistA Field Definitions | Caché Properties |
| VistA Multiples Definitions | Caché Child Classes |
| VistA Pointer Definitions | Caché Class Relationships |

## CASH\_HDR System and HIDU System

CASH\_HDR is related with the HIDU System in that HIDU utilizes basic CASH\_HDR functionality.

## Terms

* *Caché CASH\_HDR SQL Classes/Tables* – SQL Classes and Tables belonging to the CASH\_HDR System created from the *VistA Data Dictionary Files (^DD, ^DIC).*
* *CASH – Caché to SQL Mapping Tool.*
* *CASH\_HDR System* – The *CASH\_HDR System* creates Caché SQL Classes based on *VistA Data Dictionary Files (^DD, ^DIC)* that allows the SQL Classes to read Vista data in its M/native format.
* *CASH\_HDR Discovery Process* – this process deletes all *Caché CASH\_HDR SQL Classes/Tables* and then recreates them based on the *VistA Data Dictionary Files (^DD, ^DIC)*.
* *Central Source Server/HDRWEB* – distribution source.
* *Child Class* – a Sub-Class with multiple entries.
* HIDU - “HDR In-place DAT Update”.
* *HIDU Discovery Process* – this process compares the Checksums of the *Caché CASH\_HDR SQL Classes/Tables* with Checksums previously saved in the VISTA.HDSMAPCENTRAL Table.
* *HIDU* is an acronym for “HDR In-place DAT Update”. The HIDU System checks for changes between the *Caché CASH\_HDR SQL Classes/Tables Checksums* and a pre-saved list of *Checksums*. When differences are found, the affected Class or Classes are analyzed and those that should be distributed to *the VistA Database* Sites are loaded into the *Central Source Server/HDRWEB* for processing by Java Scripts. It then pushes these changes out to *VistA Database Sites*.
* HDR - Health Data Repository
* MDR – Meta-Data Repository
* *Multiple –*a Multiple is a FileMan Field with repeating values. It is stored as a sub-file within the current file/field definition, in other words, an array. This field is hierarchically related to the parent file. A Multiple represents itself in Caché SQL as a Child Class. Multi-layer Multiples are possible. For example: a FileMan File that has a Field that is a Multiple, and this Multiple may have descendent Multiples as well.
* *PLATINUM (Caché.dat) File* **-** Caché database (Caché.dat) that holds the Vista files as well as the *VistA Data Dictionary Files (^DD, ^DIC).*
* *Pointer –*a *Pointer* is a FileMan Field that links to another FileMan File.
* *Relationship*– a two-way reference between a property of one class to property of another class.
* *VistA* - Veterans Health Information Systems and Technology Architecture (VistA)
* *VistA Data Dictionary Files (^DD, ^DIC)* – a data dictionary of all files and fields with associated file numbers as well as multiples and pointers in Vista. VA FileMan stores file-related data in two main locations: ^DD and ^DIC. ^DD stores attributes of the file’s data dictionary including fields, cross-references, etc. ^DIC stores minimal data related to the file name and the location of data for the file

## Tables/Files/Globals Used

* CASH FM CLASS MAP - ^CASH (15050.11) – main interface mapping file between VistA Data Dictionary Files (^DD, ^DIC) and the Caché CASH SQL Classes/Tables
* CASH CUSTOM DATATYPES - ^CASH (15050.12)
* CASH ERRORS - ^CASH (15050.13)
* CASH PARAMETER FILE - ^CASH (15050.14)
* CASH SQL RESERVED WORDS - ^CASH (15050.19)
* *VistA Data Dictionary Files (^DD, ^DIC)* – a data dictionary of all files and fields with associated file numbers as well as multiples and pointers in Vista. VA FileMan stores file-related data in two main locations: ^DD and ^DIC. ^DD stores attributes of the file’s data dictionary including fields, cross-references, etc. ^DIC stores minimal data related to the file name and the location of data for the file
* Caché CASH SQL Classes/Tables – SQL Classes and Tables belonging to the CASH\_HDR System created from the VistA Data Dictionary Files (^DD, ^DIC)
* Storage area for override information - ^CASH(“HDR”)

# Production Processes

## CASH\_HDR consists of 3 main processes

The *CASH\_HDR System* consists of 3 main processes to build the *Caché CASH SQL Classes/Tables.*

1. **Map All Files (VISTA Method)** - This process performs:

* *CASH\_HDR Discovery Process*and
* Create/Compile Caché CASH SQL Classes/Tables for **ALL FileMan** files

1. **START^CASH Process** - This process performs:

* *CASH\_HDR Discovery Process and*
* *if the Flags contain a “C” will* *Create/Compile Caché CASH SQL Classes/Tables*

for an **Individual FileMan** filepassed by parameter

1. **CREATE^CASH Process** - This process performs:

* *Create/Compile Caché CASH SQL Classes/Tables*. It reads the previously-mapped information and compiles classes without a new discovery for an **Individual FileMan** file passed by parameter

## Differences between ALL^CASH, START^CASH and CREATE^CASH

|  |  |  |  |
| --- | --- | --- | --- |
|  | Map All Files  (VISTA Method) | START^CASH | CREATE^CASH |
| *CASH\_HDR Discovery Process* | Yes | Yes | No |
| *Create/Compile Caché CASH SQL Classes/Tables* | Yes | Yes or No - If the “C” Flag is passed to START^CASH, then CREATE^CASH will be called automatically. | Yes |
| File(s) processed | All FileMan files | Individual (passed by parameter) FileMan file | Individual (passed by parameter) FileMan file |

## Map all Vista Files – VISTA Method

**W ##class(CASH.HDR.Utility).VISTA() – no cross references**

**W ##class(CASH.HDR.Utility).VISTA(“”,1) – create cross references**

### Overview

This process goes through the *VistA Data Dictionary Files (^DD, ^DIC)* to recreate the *Caché CASH SQL Classes/Tables* so that these classes are in sync with the *VistA Data Dictionary Files (^DD, ^DIC)* .

When specified, the second optional parameter allows Caché SQL to use the VistA Cross References or Indexes for faster data access. These Indexes can been seen when editing a Class in Caché Studio by viewing the Storage.

### (CASH.HDR.Utility).VISTA() – Initial call Overview

### (CASH.Utility).All – Map all Vista Files

### ALL^CASH

### ALL^CASHC

### ALL^CASH - (CASH.HDR.Utility).VISTA() - Description by Graphic Representation

#### 

### ALL^CASH - (CASH.HDR.Utility).VISTA() Description

* Start by going through the ^CASH(15050.11) global and deletes all nodes therein.
* Then it goes through the ^DIC/^DD globals and stores the data in global ^CASH(15050.11).
* Next it adds additional needed data to ^CASH (15050.11).
* Now it goes through the newly created ^CASH (15050.11) global and deletes the associated *Caché CASH SQL Classes/Tables*.
* Step through the ^CASH (15050.11) global again and creates *Caché CASH SQL Classes/Tables* including all the associated attributes. It then to compile all classes.
* Lastly it goes through ^CASH (15050.11) and counts the classes.

# Technical Documentation

## Three primary elements of ALL^CASH - (CASH.HDR.Utility).VISTA()

* *VistA Data Dictionary Files (^DD, ^DIC)* – a data dictionary of all files and fields with associated file numbers as well as multiples and pointers in Vista. VA Fileman stores file-related data in two main locations: ^DD and ^DIC. ^DD stores attributes of the file’s data dictionary including fields, cross-references, etc. ^DIC stores minimal data related to the file name and the location of data for the file.
* *CASH FM CLASS MAP Global ^CASH (15050.11)* - This file contains the maps (or templates) used to create Caché classes for Fileman files. Each map is created by the START^CASH call, which examines the ^DIC and ^DD structures for the file. This map can be used to create many different Caché Classes, with different options (FLAGS), by using the CREATE^CASH call
* *Caché CASH\_HDR SQL Classes/Tables* – SQL Classes and Tables belonging to the CASH\_HDR System created from the *VistA Data Dictionary Files (^DD, ^DIC).*

## Key to understand ALL^CASH - (CASH.HDR.Utility).VISTA()

**The key to understanding this process is the** ^CASH (15050.11) Global. **Thisglobal is first populated from the** VistA Data Dictionary Files (^DD, ^DIC) **files and next is used to build the** CASH\_HDR Caché Classes**. It serves as a bridge between the VistA Data Dictionary and the CASH\_HDR Classes. So a complete understanding of this global is necessary.**

In the detailed description table that follows, changes to ^CASH(15050.11) will be highlighted in green.

## Detailed Description of ALL^CASH - (CASH.HDR.Utility).VISTA()

### First reads from ^DIC/^DD AND Populate ^CASH(15050.11)

|  |  |  |  |
| --- | --- | --- | --- |
| Delete all nodes from  ^CASH(15050.11) | DELALL^CASHFN11 | $ORDER thru  ^CASH(15050.11,FILE)  and delete all nodes | |
| Discover New Files | $ORDER thru ^DIC(FILE)  Calls: DISCVR^CASH0(FILLE) | Discovers new Files,  And stores in:  ^CASH(15050.11) | |
| Add data to ^CASH(15050.11,FILE) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | If Top level file then:  FILE^CASH1  Get Name, Subnm, Global, Subs | |  | | --- | | FILE^CASHF(FILE | | $$NAME^CASHN(DICNM) | | ADDFILE^CASHFN11(FILE) | | ^CASH(15050.11,4,0)="Institution^^^F^" | | | If Subfile then:  SUBFILE^CASH1 | |  | | --- | | FILE^CASHF(FLNO) | | $$NAME^CASHN(DICNM) | | $$STDNAME^CASHN(NM(1)) | | ADDFILE^CASHFN11(FILE) | |  | | | MMAP^CASH1(FILE)  Add master map | |  | | --- | | Master Map must have ID=1 and TYPE="D" (data) | | ADDMAP^CASHFN11(FILE,1,"Master","D",GLOBAL) | | ^CASH(15050.11,4,0)="Institution^^^F^"  ^CASH(15050.11,4,3,0)="^15050.113^1^1"  ^CASH(15050.11,4,3,1,0)="Master^D^DIC"  ^CASH(15050.11,4,3,"B","Master",1)=""  Note, 3rd subscript of 3 comes from 113  D is for data, ^DIC comes from GLOBAL | | F I=1:1 Q:'$D(SUBS(I)) D ADDSUBS^CASHFN11  SUBS="4," SUBS(1)=4 | | ^CASH(15050.11,4,3,1,1,0)="^15050.1131^1^1"  ^CASH(15050.11,4,3,1,1,1,0)="4^^^^"  ^CASH(15050.11,4,3,1,1,"B",4,1)=""  ^CASH(15050.11,4,3,"B","Master",1)="" | | ADDSUBS^CASHFN11 | | ^CASH(15050.11,4,3,1,1,0)="^15050.1131^2^2"  ^CASH(15050.11,4,3,1,1,1,0)="4^^^^"  ^CASH(15050.11,4,3,1,1,2,0)="#4^0^'+{L2}^^"  ^CASH(15050.11,4,3,1,1,"B",4,1)=""  ^CASH(15050.11,4,3,1,1,"B","#4",2)="" | | | NEWINX  ^CASH1(FILE)  Create array of New Type Indexes and Fields  NIX(“NIX”)  NIXF | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Loop through New-Style Index "B" cross reference for the specified FILE (Subfile)   F  S NIX=$O(^DD("IX","B",FILE,NIX))  ^DD("IX","B",4,339)=  ^DD("IX","B",4,340)=  ^DD("IX","B",4,341)=  ^DD("IX","B",4,677)=   |  |  | | --- | --- | |  | .S NIXREC=^DD("IX",NIX,0) | |  | .S NIXNM=$P(NIXREC,"^",2) - XUMFID | |  | .S MAPNM=$P(NIXREC,"^",3)  Enables lookup by coding system concatenated with ID. | |  | Loop through Fields  .F  S NIXSB=$O(^DD("IX",NIX,11.1,NIXSB))   |  |  | | --- | --- | |  | ..S NIXSBREC=^DD("IX",NIX,11.1,NIXSB,0) | |  | ..; Only map Indexes that contain just Fields  ..I $P(NIXSBREC,"^",2)'="F" S OK=0 Q | |  | ..I $P(NIXSBREC,"^",3)'=FILE S OK=0 Q | |  | ..S NIXFLD=$P(NIXSBREC,"^",4) | |  | ..; Check that the field is valid ..I NIXFLD="" S OK=0 Q | |  | ..I '$D(^DD(FILE,NIXFLD)) S OK=0 Q | |  | ..; Add field details to NIXTMP  ..S NIXTMP("NIX",NIXSB)=NIXFLD | |  | ..S NIXTMP("FLD",NIXFLD)="" | | |  | .; Merge Temp array to NIX and NIXF (passed by reference) .M NIX(NIX)=NIXTMP("NIX")  .M NIXF=NIXTMP("FLD") | | | |  |  | |  |  | | FIELDS  ^CASH2(FILE)  Add Fields  Loop thru ^DD(FILE,FIELD) | |  |  |  | | --- | --- | --- | | F  S FIELD=$O(^DD(FILE,FIELD)) | | | | |  |  | | --- | --- | |  | .FIELD^CASHF(FILE) Get Field details | |  | .S LABEL=$$STDNAME^CASHN(DDLBL) | |  | .S DESC="Field #: "\_FIELD\_" Name: "\_DDLBL | |  | .S REQ=$S(SPEC["R":1,1:0) | |  | Field .001 is a Special Case!  It returns the IEN, which will already be mapped.  .If FIELD=.001   |  |  | | --- | --- | |  | ..D LABELS(LABEL | |  | ..D ADDFLD^CASHFN11(FILE,FIELD,LABEL,0 ..D ADDDAT | |  | ..D ^CASHFN11(FILE,1,FIELD,DATALBL | | |  | .; Remove characters from SPEC not to do with data extraction  .S SPEC=$TR(SPEC,"aeAILMO'RX\*","") | |  | .; Set Output Transform flag  .S TRANS=$S(OUT'="":1,1:0) | |  | .; Check for Numeric values with Transforms (must be exposed as String/Varchar)  .I TYPE="NUMERIC",TRANS S TYPE="FREE TEXT" | |  | .; Check that Pointer is Valid, and Discover pointed to File  .; (Note: Discovery has been moved up to here to help prevent <FRAMESTACK> errors) | |  | .I TYPE="POINTER" $$DISCVR^CASH0(PFILE,PFLAGS) | |  | I $E(PIECE)="E" D ???? | | | | |  | | | | ^CASH(15050.11,4,0)="Institution^^^F^"  ^CASH(15050.11,4,3,0)="^15050.113^1^1"  ^CASH(15050.11,4,3,1,0)="Master^D^DIC"  ^CASH(15050.11,4,3,1,1,0)="^15050.1131^2^2"  ^CASH(15050.11,4,3,1,1,1,0)="4^^^^"  ^CASH(15050.11,4,3,1,1,2,0)="#4^0^'+{L2}^^"  ^CASH(15050.11,4,3,1,1,"B",4,1)=""  ^CASH(15050.11,4,3,1,1,"B","#4",2)=""  ^CASH(15050.11,4,3,"B","Master",1)="" | | | |  | Check for Duplicate fields and/or Labels  .S DUP=0,LBLUP=$$UP^XLFSTR(LABEL) | |  | .; Get SQL and Data labels  D LABELS(LABEL,.SQLLBL,.DATALBL) | |  | ; Add any Traditional Indexes  .I $D(^DD(FILE,FIELD,1)),'MULT D | |  | |  |  | | --- | --- | |  | ..S INX=$O(^DD(FILE,FIELD,1,INX)) D | |  | |  |  | | --- | --- | |  | …S INXFN=$P($G(^DD(FILE,FIELD,1,INX,0)),"^",1) | |  | ...S INXNM=$P($G(^DD(FILE,FIELD,1,INX,0)),"^",2) | |  | ...S INXTP=$P(^DD(FILE,FIELD,1,INX,0),"^",3) | |  | …S MAP=MAP+1 | |  | ...S MAPNM=LABEL\_INXNM\_"Index" | |  | ...S INDEXED=1 | |  | ...D ADDMAP^CASHFN11(FILE,MAP,MAPNM,"I",GLOBAL) | | | | |  | | --- | | ^CASH(15050.11,4,0)="Institution^^^F^"  ^CASH(15050.11,4,3,0)="^15050.113^2^2"  ^CASH(15050.11,4,3,1,0)="Master^D^DIC"  ^CASH(15050.11,4,3,1,1,0)="^15050.1131^2^2"  ^CASH(15050.11,4,3,1,1,1,0)="4^^^^"  ^CASH(15050.11,4,3,1,1,2,0)="#4^0^'+{L2}^^"  ^CASH(15050.11,4,3,1,1,"B",4,1)=""  ^CASH(15050.11,4,3,1,1,"B","#4",2)=""  ^CASH(15050.11,4,3,2,0)="NameBIndex^I^DIC"  ^CASH(15050.11,4,3,"B","Master",1)=""  ^CASH(15050.11,4,3,"B","NameBIndex",2)="" | | | | |  | |  |  | | --- | --- | |  |  | |  | |  |  | | --- | --- | |  | ...I INXFN'=FILE D  Q | |  | |  |  | | --- | --- | |  | ….F I=1:1 Q:'$D(SUBS(I)) Q:SUBS(I)=("#"\_INXFN) D ADDSUBS^CASHFN11(FILE,MAP,I,SUBS(I),"","","","") | |  | ....D ADDSUBS^CASHFN11(FILE,MAP,J,""""\_INXNM\_"""","","","","") S J=J+1 | |  | ….D ADDSUBS^CASHFN11(FILE,MAP,J,"{"\_DATALBL\_"}","","",FIELD,ALTEXP) S J=J+1 | |  | ....F I=I:1 Q:'$D(SUBS(I)) | |  | ….I $E(SUBS(I))="#" D ADDSUBS^CASHFN11(FILE,MAP,J,SUBS(I),"","","","") S J=J+1 | |  | ....D ADDSUBS^CASHFN11(FILE,MAP,J,UID,"","","","") | | |  | ...F I=1:1 Q:'$D(SUBS(I)) D ADDSUBS^CASHFN11(FILE,MAP,I,SUBS(I),"","","","") | |  | ...D ADDSUBS^CASHFN11(FILE,MAP,I,""""\_INXNM\_"""","","","","") | |  | ...D ADDSUBS^CASHFN11(FILE,MAP,I+1,"{"\_DATALBL\_"}","","",FIELD,ALTEXP) | |  | ...D ADDSUBS^CASHFN11(FILE,MAP,I+2,UID,"","","","") | |  |  | | | |  | | | | I $D(^DD(FILE,FIELD,1)),'MULT D – continued below | | | | I MULT D MULT^CASH3(FILE,FIELD – continued below | | | | I TYPE="DATE/TIME" D DT^CASH3(FILE,FIELD – continued below | | | | I TYPE="NUMERIC" D NUM^CASH3(FILE,FIELD – continued below | | | | I TYPE="SET" D SET^CASH3(FILE,FIELD – continued below | | | | .I TYPE="FREE TEXT" D FREE^CASH3(FILE,FIELD – continued below | | | | I TYPE="WORD-PROCESSING" D WP^CASH3(FILE,FIELD – continued below | | | | I TYPE="COMPUTED" D COMP^CASH3(FILE,FIELD – continued below | | | | I TYPE="POINTER" D PNTR^CASH3(FILE,FIELD – continued below | | | | I TYPE="VARIABLE-POINTER" D VP^CASH3(FILE,FIELD – continued below | | | | I TYPE="MUMPS" D MUMPS^CASH3(FILE,FIELD – continued below | | | |  | | | | | NEWINXU^CASH1(FILE)  Create New Type Indexes | |  | | --- | | ADDMAP^CASHFN11(FILE) | | ADDSUBS^CASHFN11(FILE) | | | | |
| Get Class Name and  Delete Caché Class | $ORDER thru  ^CASH(15050.11,FILE)  Calls: $$GETCLNM^CASHCN | | CLDEL^CASHCU(CLNAME) |
| Get Class Name | $ORDER THRU  ^CASH(15050.11,FILE)  Calls: $$GETCLNM^CASHCN  Return Class Name for FM File # | | $$CREATE^CASHC(FILE) – next table |

#### FIELDS^CASH2(FILE) – continued from above

|  |  |
| --- | --- |
| I $D(^DD(FILE,FIELD,1)),'MULT D |  |
| I MULT  D MULT^CASH3(FILE,FIELD |  |
| I TYPE="DATE/TIME"  D DT^CASH3(FILE,FIELD |  |
| I TYPE="NUMERIC"  D NUM^CASH3(FILE,FIELD |  |
| I TYPE="SET" D SET^CASH3(FILE,FIELD |  |
| .I TYPE="FREE TEXT"  D FREE^CASH3(FILE,FIELD | |  |  | | --- | --- | | S PTYPE="CASH.FileMan.String" |  | | D ADDFLD^CASHFN11  (FILE,FIELD,LABEL,0,"","",  DESC,"",0,REQ,"",0,PTYPE,"D",SQLLBL) | |  | | --- | | S CASHIEN(1)=FIELD - .01 | | S IENS="+1,"\_FILE\_"," - “+1,4,” | | S CASHFDA(15050.111,IENS,.01)=NAME  ; Field Name – “Name” | | S CASHFDA(15050.111,IENS,.02)=CALC  ; Calculated (1 or 0) - 0 | | S CASHFDA(15050.111,IENS,.03)=CARD  ; Cardinality ("",one,many,parent,child) – “” | | S CASHFDA(15050.111,IENS,.04)=COLL  ; Collection ("",array,list) - “” | | S CASHFDA(15050.111,IENS,.05)=INV  ; Inverse Property (Relationships) – “” | | S CASHFDA(15050.111,IENS,.06)=REL  ; Relationship (1 or 0) - 0 | | S CASHFDA(15050.111,IENS,.07)=REQ  ; Required (1 or 0) - 1 | | S CASHFDA(15050.111,IENS,.08)=SQLCOMP  ; SQL Computed (1 or 0) - 0 | | S CASHFDA(15050.111,IENS,.09)=TYPE  ; Cache Datatype - "CASH.FileMan.String" | | S:DESC'="" CASHFDA(15050.111,IENS,.11)=DESC  ; Description - Field #: .01 Name: NAME | | S:SQLCODE'=""  CASHFDA(15050.111,IENS,.21)=SQLCODE  ; SQL Code for Computed Field | | S CASHFDA(15050.111,IENS,.31)=CAT ; Field Category (data,multiple,child,pointer) - “D” | | S CASHFDA(15050.111,IENS,.32)=SQLNM  ; SQL Field Name |   ^CASH(15050.11,4,1,0)=^15050.111^.01^1  ^CASH(15050.11,4,1,.01,0)=Name^0^^^^0^1^0^CASH.FileMan.String  ^CASH(15050.11,4,1,.01,1)=Field #: .01 Name: NAME  ^CASH(15050.11,4,1,.01,3)=D^  ^CASH(15050.11,4,1,"B","Name",.01)=  ^CASH(15050.11,4,1,"C","D",.01)=  ^CASH(15050.11,4,1,"U","NAME",.01)= | | ; Add Parameters  S MAX=+$P($P(INP,"K:$L(X)>",2),"!",1)  S MIN=+$P($P(INP,"!($L(X)<",2),")",1) |  | | D ADDPARAM  ^CASHFN11(FILE,FIELD,1,"FILE",FILE) |  | | D ADDPARAM  ^CASHFN11(FILE,FIELD,2,"FIELD",FIELD) |  | | D ADDPARAM  ^CASHFN11(FILE,FIELD,3,"TRANSFORM",TRANS) |  | | S I=4 |  | | I MAX'=0 D ADDPARAM  ^CASHFN11(FILE,FIELD,I,"MAXLEN",MAX) S I=I+1 |  | | I MIN'=0 D ADDPARAM  ^CASHFN11(FILE,FIELD,I,"MINLEN",MIN) S I=I+1 |  | | I INDEXED D ADDPARAM  ^CASHFN11(FILE,FIELD,I,"COLLATION","EXACT") |  | | ; Add to Master Map Data  D ADDDATA  ^CASHFN11  (FILE,1,FIELD,DATALBL,  NODE,PIECE,0,RCODE) |  | |
|  |  |
| I TYPE="WORD-PROCESSING"  D WP^CASH3(FILE,FIELD |  |
| I TYPE="COMPUTED"   D COMP^CASH3(FILE,FIELD |  |
| I TYPE="POINTER"  D PNTR^CASH3(FILE,FIELD  Add Reference to Pointed to Class/Table | |  |  | | --- | --- | | Pointer Files  S PFILE=+$P(SPEC,"P",2) Q:'PFILE |  | | ; Add Field  D ADDFLD^CASHFN11  (FILE,FIELD,LABEL,0,"",""  ,DESC,"",0,REQ,"",0,"CASH.  FileMan.Pointer","P",SQLLBL) |  | | ; Add Parameters  D ADDPARAM^CASHFN11  (FILE,FIELD,1,"PFILE",PFILE)  D ADDPARAM^CASHFN11  (FILE,FIELD,2,"PFIELD",".01") |  | | ; Add to Master Map Data  D ADDDATA^CASHFN11  (FILE,1,FIELD,DATALBL,  NODE,PIECE,1,RCODE) |  | |  |  | |  |  | |
| I TYPE="VARIABLE-POINTER"  D VP^CASH3(FILE,FIELD,L |  |
| I TYPE="MUMPS"  D MUMPS^CASH3(FILE,FIELD |  |

#### This section reads from ^CASH(15050.11) and creates the Classes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Create CASH Class  $$CREATE^CASHC(FILE) | |  |  | | --- | --- | |  |  | | |  | | --- | | ^CASH(15050.11,FILE) | | D GETNAMES^CASHCN  ^CASH(15050.11,FILE,0)  I FLAGS["h" D NAMES^CASHHDRO(FILE  Override Names (^CASH("HDR","File",FILE)) | | D CLDEL^CASHCU(CLNAME)  Delete Class if it exists | | D MAXVALS^CASHCU(FILE) |   $$CLASS^CASHC0 Create Class   |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%New() | | ##CLASS(%Dictionary.ParameterDefinition).%New() | | ##CLASS(%Dictionary.PropertyDefinition).%New() | | ##CLASS(%Dictionary.IndexDefinition).%New() | |  | | (Create subclasses)  F  S FIELD=$O(^CASH(15050.11,FILE,1,"C","M",FIELD))  . $$CREATE^CASHC | |  |  | | --- | --- | | D GETNAMES^CASHCN  ^CASH(15050.11,FILE,0) | | | D CLDEL^CASHCU(CLNAME)  Delete Class if it exists | | D MAXVALS^CASHCU(FILE) | | |   $$CLASS^CASHC0 Create SubClasses   |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%New() | | ##CLASS(%Dictionary.ParameterDefinition).%New() | | ##CLASS(%Dictionary.PropertyDefinition).%New() | | ##CLASS(%Dictionary.IndexDefinition).%New() | | | $$CREATE^CASHC  (recursion) | |  |  | | --- | --- | | D GETNAMES^CASHCN  ^CASH(15050.11,FILE,0) | | | D CLDEL^CASHCU(CLNAME)  Delete Class if it exists | | D MAXVALS^CASHCU(FILE) | | |   $$CLASS^CASHC0 Create Pointed to Class   |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%New() | | ##CLASS(%Dictionary.ParameterDefinition).%New() | | ##CLASS(%Dictionary.PropertyDefinition).%New() | | ##CLASS(%Dictionary.IndexDefinition).%New() | | | $$PROPS^CASHC1 Properties | |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%OpenId | | ##CLASS(%Dictionary.PropertyDefinition).%New() | | $$IENS^CASHHDRO(FILE) | | - D PROPERTY^CASHHDRO | | ##CLASS(%Dictionary.PropertyDefinition).%New() | | | $$METHODS^CASHC2 Methods | |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%OpenId(CLNAME) | | | $$TRIGGERS^CASHC3 Triggers |  | | $$STORAGE^CASHC4 Storage | |  | | --- | | ##CLASS(%Dictionary.ClassDefinition).%OpenId(CLNAME) | | ##CLASS(%Dictionary.StorageDefinition).%New() | | MAPS | | F I=1:1 Q:'$D(^CASH(15050.11,FILE,3,I)) | | MAPNODE0=$G(^CASH(15050.11,FILE,3,I,0)) | | ##CLASS(%Dictionary.StorageSQLMapDefinition).%New() | | SUBS | | F J=1:1 Q:'$D(^CASH(15050.11,FILE,3,I,1,J)) | | SUBNODE0=$G(^CASH(15050.11,FILE,3,I,1,J,0)) | | I FLAGS["h",FLD D EXP^CASHHDRO(FILE | | | $$REL^CASHC1 Continuation Class |  | | $$EXTRA^CASHHDRE Extras (Variable Pointers?) | I '$D(^CASH("HDR","File",FILE,"Node")),'$D(^CASH("HDR","File",FILE,"VP")) Q | | | |
| Merge and Compile Classes | | M LIST=^TMP("CASHC",$J,  "COMPILE-LIST") | CLCMPLST^CASHCU(.LIST | |
| Get Class Name & Count | | $ORDER THRU  ^CASH(15050.11,FILE) | $$GETCLNM^CASHCN | |

# Globals/Tables/Files

## ^CASH (“HDR”) Global

Override global – used in conjunction with the “h” flag.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Global Node and Subscripts | Piece1 | Piece2 | Piece3 | Piece4 |
| ^CASH("HDR","ClassName",Name) | FileNum |  |  |  |
| ^CASH("HDR","File",FileNum) | NAME | SQLNM |  |  |
| ^CASH("HDR","Names",FileNum,IEN Name) | Datatype  Number |  |  |  |
| ^CASH("HDR","File",FileNum,"Fld",FieldNum) | LABEL/NAME | SQLNM | Datatypes | Inverse  Relationship |
| ^CASH("HDR","File",FileNum,"Node",NodeNum) | LABEL | SQLNM | Node |  |
| ^CASH("HDR","File",FileNum,"VP",Number)  VP – variable pointers | LABEL | SQLNM | PFILE  (Pointer  File) |  |

## CASH FM CLASS MAP Global ^CASH (15050.11)

This file contains the maps (or templates) used to create Caché classes for Fileman files. Each map is created by the START^CASH call, which examines the ^DIC and ^DD structures for the file. This map can be used to create many different Caché Classes, with different options (FLAGS), by using the CREATE^CASH call.

## CASH CUSTOM DATATYPES Global ^CASH (15050.12)

This file contains the maps (or templates) used to create the custom datatype Classes that support the CASH tool.

## CASH ERRORS Global ^CASH (15050.13)

This file contains any error messages generated by the compile process, CREATE^CASH.

## CASH PARAMETER FILE Global ^CASH (15050.14)

This file contains Parameters that can be used during the compile process.

## CASH LOCALIZATION FILE Global ^CASH (15050.17)

## CASH SQL RESERVED WORDS Global ^CASH (15050.19)

This file contains a list of SQL Reserved Words. When the discovery process (START^CASH) assigns names, each one is checked against this list. If a match is found, an alternate name is stored for use in a SQL context. The alternate name is the original name with an underscore prefix (e.g. TABLE would be stored as \_TABLE).

## CASH FM CLASS MAP Global ^CASH (15050.11)

This file contains the maps (or templates) used to create Caché classes for Fileman files. Each map is created by the START^CASH call, which examines the ^DIC and ^DD structures. This map can be used to create many different Caché Classes, with different options (FLAGS), by using the CREATE^CASH call.

* + 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) - Detail***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | File/Field | Node;  Piece | Data  Type | Definition | Description |
| ^DD(15050.11,.01,1,1,0)=15050.11^B  ^DD(15050.11,.01,1,1,1)=S ^CASH(15050.11,"B",$E(X,1,30),DA)=""  ^DD(15050.11,.01,1,1,2)=K ^CASH(15050.11,"B",$E(X,1,30),DA)  ^DD(15050.11,.01,1,2,0)=15050.11^U^MUMPS  ^DD(15050.11,.01,1,2,1)=S ^CASH(15050.11,"U",$E($$UP^XLFSTR(X),1,30),DA)=""  ^DD(15050.11,.01,1,2,2)=K ^CASH(15050.11,"U",$E($$UP^XLFSTR(X),1,30),DA) | | | | | |
| NAME | 15050.11,.01 | 0;1 | Free  Text | 3-30  Required | Standardized Name for the Class to be created. If this Class is based on a top-level file, the name is taken from ^DIC. For Sub-files or WP Fields the name is a Standardized compound of this file's ^DD name, prefixed by the Parent file names. Note: The Standardizing process makes all words title case, then removes all spaces and other punctuation. The names are then shortened to 25 characters or less |
|  |  |  |  |  | ^CASH(15050.11,4,0)=Institution^^^F^ |
| SUB  NAME | 15050.11,.02 | 0;2 | Free  Text | 3-30  Required | Standardized name of this element only (not prefixed by parent file names), if this Class is based on a sub-file or WP Field. This field is left blank for top-level files. SUB NAME is used to name the Class if the “N” flag is used. |
|  |  |  |  |  | ^CASH(15050.11,4.0051,0)="Md5SgntrFileSbflNmbr^FileSubfileNumber^^S^1" |
| SQL  NAME | 15050.11,.03 | 0;3 | Free  Text | 3-30  Required | The name to be used for the SQL table (if different from NAME). SQL NAME must be specified if NAME is a reserved word in SQL. |
| FILE  TYPE | 15050.11,.04 | 0;4 | Set | F for File  S for Sub-File  W for WP Sub-  File  Required | Type of the file this Class is based on (top-level, sub-file or WP Field).  Used by the compiler to generate appropriate update code. |
|  |  |  |  |  | ^CASH(15050.11,4,0)=Institution^^^F^ |
|  |  |  |  |  | ^CASH(15050.11,4.0051,0)="Md5SgntrFileSbflNmbr^FileSubfileNumber^^S^1 |
| PARENT  FIELD | 15050.11,.05 | 0;5 | Number | Between  .0000001 and  99999999. | If this Class is based on a sub-file or WP Field, this field points to the related field in the parent file. |
|  |  |  |  |  | ^CASH(15050.11,4.0051,0)="Md5SgntrFileSbflNmbr^FileSubfileNumber^^S^1 |

* + - 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail - Properties***

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PROPERTIES | | 15050.11,1 | | 1;0 | Multiple | | | 1-30 | | Multiple containing Property details for the class. | | |
| Name | | File/Field | | | Node;  Piece | Data  Type | | Definition | | Description | |
| ^DD(15050.111,.01,1,1,0)=15050.111^B  ^DD(15050.111,.01,1,1,1)=S ^CASH(15050.11,DA(1),1,"B",$E(X,1,30),DA)=""  ^DD(15050.111,.01,1,1,2)=K ^CASH(15050.11,DA(1),1,"B",$E(X,1,30),DA)  ^DD(15050.111,.01,1,2,0)=15050.111^U^MUMPS  ^DD(15050.111,.01,1,2,1)=S ^CASH(15050.11,DA(1),1,"U",$E($$UP^XLFSTR(X),1,30),DA)=""  ^DD(15050.111,.01,1,2,2)=K ^CASH(15050.11,DA(1),1,"U",$E($$UP^XLFSTR(X),1,30),DA) | | | | | | | | | | | |
| NAME | | 15050.111,.01 | | | 0;1 | Free  Text | | 1-30  Required | | | Standardized name for the Property to be created. |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| CALCULATED | | 15050.111,.02 | | | 0;2 | Set | | 0 for No  1 for Yes  Required | | | Flag identifying whether or not the Property is calculated. |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| CARDINALITY | | 15050.111,.03 | | | 0;3 | Set | | O for One  M for Many  P for Parent  C for Children | | | Cardinality of a relationship type Property (one-to-many or parent-child) |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| COLLECTION | | 15050.111;.04 | | | 0;4 | Set | | L for List  B for Binary  Stream  C for Character  Stream | | | Type of collection Property. |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| INVERSE | | 15050.111,.05 | | | 0;5 | Free  Text | | 1-30 | | | Inverse Class Name for a Relationship type Property. |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| RELATIONSHIP | | 15050.111,.06 | | | 0;6 | Set | | 0 for No  1 for Yes | | | Is the Property a relationship? |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| REQUIRED | | 15050.111,.07 | | | 0;7 | Set | | 0 for No  1 for Yes  Required | | | Is the Property a required? |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| SQL  COMPUTER | | 15050.111,.08 | | | 0;8 | Set | | 0 for No  1 for Yes | | | Is the Property a SQL Computed field? |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| TYPE | | 15050.111,.09 | | | 0;9 | Free  Text | | 1-30  Required | | | The data type of the Property |
|  | |  | | |  | ^CASH(15050.11,4.005,1,.01,0)=Md5CodeSystem^0^^^^0^1^0^CASH.FileMan.String | | | | | |
| DESCRIPTION | | 15050.111,.11 | | | 1;1 | Free  Text | | 1-80 | | | The description of the Property, used in the Class  definition. |
|  | |  | | |  |  | | | | | |
| SQL COMPUTE CODE | | 15050.111,.21 | | | 2;E1,245 | Free  Text | |  | | | SQL compute code for a SQL computed property. |
|  | |  | | |  |  | | | | | |
| FIELD  CATEGORY | | 15050.111,.31 | | | 3;1 | Set | | C for child  D for data  M for multiple  P for pointer  T for transient  W for multiple (wp) | | | Field category of the Property, used internally by the compiler |
|  | |  | | |  |  | | | | | |
| SQL FIELD  NAME | | 15050.111,.32 | | | 3;2 | Free  Text | | 3-30 | | | The SQL Field Name for the  property (if different from Name) |
|  | |  | | |  |  | | | | | |

* + - * 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail – Properties - Parameters***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PARAMETERS | | 15050.111,1 | | 1;0 | | Multiple | |  | | Multiple containing Property parameters for the Class | |
| Name | | File/Field | | Loc | | Data  Type | | Definition | | Description |
| ^DD(15050.1111,.01,1,1,0)=15050.1111^B  ^DD(15050.1111,.01,1,1,1)=S ^CASH(15050.11,DA(2),1,DA(1),1,"B",$E(X,1,30),DA)=""  ^DD(15050.1111,.01,1,1,2)=K ^CASH(15050.11,DA(2),1,DA(1),1,"B",$E(X,1,30),DA) | | | | | | | | | | |
| PARAMETER  NAME | | 15050.1111,.01 | | 0;1 | | Free  Text | | 1-30  Required | | The Name of the Parameter |
|  | |  | |  | | ^CASH(15050.11,4.011,1,.01,1,0)=^15050.1111^6^6  ^CASH(15050.11,4.011,1,.01,1,1,0)=FILE^4.011  ^CASH(15050.11,4.011,1,.01,1,2,0)=FIELD^.01  ^CASH(15050.11,4.011,1,.01,1,3,0)=TRANSFORM^0  ^CASH(15050.11,4.011,1,.01,1,4,0)=MAXLEN^40  ^CASH(15050.11,4.011,1,.01,1,5,0)=MINLEN^1 | | | | |
| PARAMETER  VALUE | | 15050.1111,.02 | | 0;2 | | Free  Text | | 1-30 | | The parameter’s default value |
|  | |  | |  | | ^CASH(15050.11,4.011,1,.01,1,0)=^15050.1111^6^6  ^CASH(15050.11,4.011,1,.01,1,1,0)=FILE^4.011  ^CASH(15050.11,4.011,1,.01,1,2,0)=FIELD^.01  ^CASH(15050.11,4.011,1,.01,1,3,0)=TRANSFORM^0  ^CASH(15050.11,4.011,1,.01,1,4,0)=MAXLEN^40  ^CASH(15050.11,4.011,1,.01,1,5,0)=MINLEN^1 | | | | |
| STRING | | 15050.1111,1 | | 1;0 | | M Code | |  | | Calculated field to  display the parameter detail |
|  | |  | |  | |  | | | | |

* + - 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail - Methods***

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| METHOD | | 15050.11,,2 | | 2;0 | Multiple | | | 3-30  Required | | | Multiple containing Method details for the Class. | |
| NAME | | File/Field | | | Loc | | | Data  Type | Definition | | Description | |
| ^DD(15050.112,.01,1,1,0)=15050.112^B  ^DD(15050.112,.01,1,1,1)=S ^CASH(15050.11,DA(1),2,"B",$E(X,1,30),DA)=""  ^DD(15050.112,.01,1,1,2)=K ^CASH(15050.11,DA(1),2,"B",$E(X,1,30),DA) | | | | | | | | | | | | |
| METHOD  NAME | | 15050.112,.01 | | | 0;1 | | | Free  Text | 1-30  Required | | The Method Name | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| CLASS  METHOD | | 15050.112,.02 | | | 0;2 | | | Set | 0 for No  1 for Yes  Required | | The Class Method | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| CODE MODE | | 15050.112,.03 | | | 0;3 | | | Set | A for Call  C for Code  E for Expression  G for Generator  O for  Objectgenerator | | How the Method code is implemented (call, code, expression, generator, or object generator). Most Methods used by this tool will be a simple expression or a list of lines (code). The more complex datatype methods often use Method generators. | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| PRIVATE | | 15050.112,.04 | | | 0;4 | | | Set | 0 for No  1 for Yes | | Is the Method private? | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| RETURN TYPE | | 15050.112,.05 | | | 0;5 | | | Free  Text | 1-30 | | The datatype of the return  value (if applicable). | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| FIELD NUMBER | | 15050.112,.06 | | | 0;6 | | | Number | Between  .0000001 and  99999999 | | The FileMan field number associated with this Method (if applicable). If the Field being mapped is excluded from compilation, the Method will not be created. | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,0)=Level1NameGet^0^E^0^CASH.FileMan.String | | | | |
| DISCRIPTION | | 15050.112,.11 | | | 1;1 | | | Free  Text | 1-80 | | The description of the Method, used in the Class  definition. | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,1)=Get() Method for Computed Field #: 20 Name: LEVEL 1 NAME | | | | |
| FORMAL  SPEC | | 15050.112,.21 | | | 2;1 | | | Free  Text | 1-80 | | The formal specification for this Method (a list of  Parameters and their Datatype) | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,2)= | | | | |
| PUBLIC  LIST | | 15050.112,.31 | | | 3;1 | | | Free  Text | 1-80 | | A list of public variables to be declared for this Method. This is only relevant for ProcedureBlock(=True) Methods, where variables need to be passed to called non-ProcedureBlock code. | |
|  | |  | | | |  | | ^CASH(15050.11,4.2,2,20,3)= | | | | |
| CODE | | 15050.112,1 | | |  | | |  |  | | Multiple containing the implementation code for the Method | |

* + - 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail - Maps***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MAPS | | | 15050.11,3 | | | 3;0 | Multiple | | |  | | | | Multiple containing storage map details for the Class | |
| NAME | | | File/Field | | | | Loc | | | Data  Type | | Definition | | Description | |
| ^DD(15050.113,.01,1,1,0)=15050.113^B  ^DD(15050.113,.01,1,1,1)=S ^CASH(15050.11,DA(1),3,"B",$E(X,1,30),DA)=""  ^DD(15050.113,.01,1,1,2)=K ^CASH(15050.11,DA(1),3,"B",$E(X,1,30),DA) | | | | | | | | | | | | | | | |
| MAP  NAME | | | 15050.113,.01 | | | | 0;1 | | | Free  Text | | 1-30  Required | | Standardized name for the storage map to be created. Only one storage map per  Class is defined by the CASH tool. | |
|  | | |  | | | |  | | | CASH(15050.11,3.704,3,1,0)=Master^D^XMB | | | | |
| TYPE | | | 15050.113,.02 | | | | 0;2 | | | Set | | D for Data  I for Index  B for Bitmap  E for Bitmapextent | | The map type (data, index, bitmap, or bitmapextent). This tool will generate one  Data Map per Class in the default "FileMan" Storage Map definition. All other Maps that are defined for this Class (if any) will be Indexes  (of Type Index, Bitmap or Bitmapextent | |
|  | | |  | | | |  | | | CASH(15050.11,3.704,3,1,0)=Master^D^XMB | | | | |
| GLOBAL  NAME | | | 15050.113,.03 | | | | 0;3 | | | Free  Text | | 1-8  Required | | The global that will be referenced by this storage map | |
|  | | |  | | | |  | | | CASH(15050.11,3.704,3,1,0)=Master^D^XMB | | | | |
|  | | |  | | | |  | | |  | |  | |  | |

* + - * 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail – Maps - Subscripts***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SUBSCRIPTS | | 15050.113,1 | | 1;0 | | Multiple | |  | | Multiple containing the subscripts for this storage map | |
| Name | | File/Field | | Node;  Piece | | Data  Type | | Definition | | Description |
| ^DD(15050.1131,.04,1,1,0)=15050.1131^C  ^DD(15050.1131,.04,1,1,1)=S ^CASH(15050.11,DA(2),3,DA(1),1,"C",$E(X,1,30),DA)=""  ^DD(15050.1131,.04,1,1,2)=K ^CASH(15050.11,DA(2),3,DA(1),1,"C",$E(X,1,30),DA)  ^DD(15050.1131,.04,1,2,0)=15050.113^F  ^DD(15050.1131,.04,1,2,1)=S ^CASH(15050.11,DA(2),3,"F",$E(X,1,30),DA(1),DA)=""  ^DD(15050.1131,.04,1,2,2)=K ^CASH(15050.11,DA(2),3,"F",$E(X,1,30),DA(1),DA) | | | | | | | | | | |
| EXPRESSION | | 15050.1131,.01 | | 0;1 | | Free  Text | | 1-30  Required | | The expression to be used for this subscript in the storage map. This expression can be a string (e.g. "5" or "B"), a reference to a field with curly braces (e.g. "{Expression}"), or a reference to a FileMan file with a preceding "#" (e.g. "#5"). The expression refers to the STATE file Class. The name is not stored but rather is resolved at runtime, depending on the FLAGS and PACKAGE variables passed |
|  | |  | |  | |  | | | | | |
| LOOP INIT  VALUE | | 15050.1131,.02 | | 0;2 | | Free  Text | | 1-10 | | The loop initialization value used by the  storage map for this subscript, if applicable.  FileMan data maps include a zero node that contains no relevant data for an individual row in Caché. Setting the LoopInitValue=0 instead of the default (null) causes this zero  node to be skipped during creation of the Caché Class |
|  | |  | |  | |  | | | | | |
| STOP EXPRESSION | | 15050.1131,.03 | | 0;3 | | Free  Text | | 1-10 | | The stop expression used by the storage map for this subscript, if applicable. FileMan data maps often contain index subscripts held at the same logical level as data. The index subscripts are nonnumeric, e.g. "B,” whereas the data-level subscripts are always numeric. Using a stop expression of "'+{Lx}" (where {Lx} is the appropriate subscript level variable) ensures that the loop quits when  there are no more data rows. |
|  | |  | |  | |  | | | | | |
| FIELD NUMBER | | 15050.1131,.04 | | 0;4 | | Number | | Between  .0000001 and  99999999. | | The FileMan field number that this expression is associated with, where applicable. This field is used to identify a subset of FileMan Fields to map. When a Class is being compiled, only the specified subset of the fields is included in the Class |
|  | |  | |  | |  | | | | | |
| ALTERNATE  EXPRESSION | | 15050.1131,.05 | | 0;5 | | Free  Text | | 3-30 | | An alternate expression for this expression, where applicable. This field is used for indexed Pointer fields. The alternate expression will usually have "ID" appended, e.g. "{State}" and "{StateID}". If  FLAGS contains "E," then a map will be created for this Class, using the default expression. If FLAGS contains "P" a map will be created using the alternate expression. If FLAGS contains both "E" and "P", then both maps will be created; but, if FLAGS contains neither "E" nor "P" this storage map will not be created at all. |
|  | |  | |  | |  | | | | | |

* + - * 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail – Maps - Data***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DATA | | 15050.113,2 | | 2;0 | | Multiple | |  | | Multiple containing data elements for this storage map, where applicable. Index maps will not usually contain any data elements | |
| Name | | File/Field | | Loc | | Data  Type | | Definition | | Description |
| ^DD(15050.1132,.01,1,1,0)=15050.1132^B  ^DD(15050.1132,.01,1,1,1)=S ^CASH(15050.11,DA(2),3,DA(1),2,"B",$E(X,1,30),DA)=""  ^DD(15050.1132,.01,1,1,2)=K ^CASH(15050.11,DA(2),3,DA(1),2,"B",$E(X,1,30),DA) | | | | | | | | | | |
| NAME | | 15050.1132,.01 | | 0;1 | | Free  Text | | 1-30  Required | | Standardized name for the data element. This must match a property  name defined in file #15050.111 |
| NODE | | 15050.1132,.02 | | 0;2 | | Free  Text | | 1-30  Required | | The global node that contains this data element. A FileMan global reference is defined by the global name, the subscripts and this node, e.g. the STATE file (#5) is  stored in ^DIC (Global Name), with two subscripts: 5 and the IEN, and the data is all contained in the '0' (zero) node. The full global reference would be ^DIC(5,IEN,0) |
| PIECE | | 15050.1132,.03 | | 0;3 | | Number | | Between 1-999,  no decimal  digits. | | The piece of the global node that contains this data element. It is assumed that the delimiter will be "^" and the $Piece function will be used to extract the  appropriate sub-string. |
| POINTER FLAG | | 15050.1132,.04 | | 0;4 | | Number | | 0 or 1  Required | | Is the data element a Pointer? |
| RETRIEVAL  CODE | | 15050.1132,.11 | | 1;E1,245 | | Free Text | |  | | The retrieval code for this data element, where applicable. This field is used when NODE and PIECE are not defined and custom code must be used to extract the data |
|  | |  | |  | |  | |  | |  |

* + - 1. ***CASH FM CLASS MAP Global ^CASH (15050.11) – Detail – Sub-Files***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SUB FILES | | 15050.11,4 | | 4;0 | Pointer Multiple | |  | | | Multiple containing sub-files for the Class | |
| NAME | | File/Field | | | Loc | | Data  Type | Definition | | Description | |
| ^DD(15050.114,.01,1,1,1)=S ^CASH(15050.11,DA(1),4,"B",$E(X,1,30),DA)=""  ^DD(15050.114,.01,1,1,2)=K ^CASH(15050.11,DA(1),4,"B",$E(X,1,30),DA)  ^DD(15050.114,.01,1,2,0)=15050.11^S  ^DD(15050.114,.01,1,2,1)=S ^CASH(15050.11,"S",$E(X,1,30),DA(1),DA)=""  ^DD(15050.114,.01,1,2,2)=K ^CASH(15050.11,"S",$E(X,1,30),DA(1),DA) | | | | | | | | | | | |
| SUB FILE | | 15050.114,.01 | | | 0;1 | | Pointer | Pointer to Cash  FM Class Map  File (#15050.11). | | Pointer to the sub-file | |

|  |  |
| --- | --- |
| ^CASH(15050.11,FILE,1,”C”,”D”,FIELD) | Data |
| ^CASH(15050.11,FILE,1,”C”,”C”,FIELD) |  |
| ^CASH(15050.11,FILE,1,”C”,”M”,FIELD) | Multiple |
| ^CASH(15050.11,FILE,1,”C”,”P”,FIELD) | Pointer |
| ^CASH(15050.11,FILE,1,”C”,”T”,FIELD) | Transient |
| ^CASH(15050.11,FILE,1,”C”,”W”,FIELD) | Word Processing |
| ^CASH(15050.11,FILE,3,”F” | Indexed Field |
| ^CASH(15050.11,FILE,3,”U” | Uppercase Index |
| ^CASH(15050.11,FILE,3,”B” | Mapnum |

1. ***FLAGS***
   1. ***Compile - “C”***

Create & Compile the Caché classes after the file(s) have been Discovered. If “C” is not passed, all parameters other than FILE and the “V” Flag are ignored by START^CASH.

* 1. ***Description - "D"***

Adds the full field description from ^DD to the Property Description. This data does not need to be stored in file #15050.11, as it is just copied directly at compile time, and is therefore a compile only option (FLAGS[“C”) .

* 1. ***Expand - “E”***

Pointers will generate a Computed Property that expands the .01 field (by default) in the pointed to file.

* 1. ***Force - “F”***

Force the creation and compilation of a Class even if it already exists. The old version of the Class will be deleted.

* 1. ***Override Names - “h”***

Override items from (^CASH("HDR",

* 1. ***Simple IDs - “I”***

Use a simple ID rather than one generated from the Class name “IEN” will be used, unless an alternative ID is passed (e.g. ID=“RowID”).

* 1. ***Loose Validation - “L”***

This flag relaxes the constraints placed on Properties, so exporting data to a 3rd party SQL database should be easier. You are less likely to get import errors when the FileMan data does not meet its own constraints (e.g. nulls in a required field, or an invalid date in a Date/Time field).

* 1. ***Multiples - “M”***

Generate Sub-Classes for Multiple fields and create the appropriate Parent-Child Relationships.

* 1. ***Simple Names - “N”***

Use simpler names, i.e. the file name concatenated with file #. Multiples won’t get parent and grandparent names prefixed.

* 1. ***Pointers - “P”***

Generate Classes for pointed to files and create the appropriate Foreign Keys.

* 1. ***SQL Only Compile - “Q”***

Only tables are compiled, not full classes. Caché Classes generally provide multiple types of access (e.g. SQL and Object). Using the “Q” flag restricts access to SQL only, so objects cannot be instantiated and classmethods cannot be called. This flag is useful if you know you only need SQL access, or want to restrict access to SQL-only. [NOTE: Can be used with the “E” flag]

* 1. ***Recursive - “R”***

Multiples and Pointers will also generate Sub-Classes and Classes for their Multiples and Pointers.

If neither “R” nor “r” are passed, “P” will be stripped from FLAGS when calling CREATE^CASH to create Sub-Classes, and both “M” and “P” will be removed when creating Pointer Classes. If “R” is passed, “M” and “P” will remain in FLAGS for all Sub-Classes and Pointer Classes.

WARNING: Using “R” can generate lots of Classes.

* 1. ***Partially Recursive - “r”***

Multiples will generate Classes for any Pointer Fields, but those Pointers will not create Sub-Classes or Classes for Multiples or Pointers that they contain. If “r” is passed “P” will remain in FLAGS for Sub-Classes but both “M” and “P” will be stripped from Pointer Classes. If “R” is passed it will override “r”.

* 1. ***SOAP Web Services - “S”***

Create a new sister Class inherited from %SOAP.WebService. This Class contains SQL Stored Procedures generated for each index in the main Class (plus the default ByID Procedure). These Queries are also exposed as Web Methods. You can invoke a web test page with the following url: http://{servername}/csp/{namespace}/{package}.{class}WS.cls

You can view the WSDL Service Description by appending “?WSDL=1” to this url. For example:

* http://localhost/csp/vista/User.PatientWS.cls
* http://localhost/csp/vista/User.PatientWS.cls?WSDL=1
  1. ***Verbose - “V”***

The default is Silent Mode (though the Caché compile messages will print to screen either way).

* 1. ***Web Page - “W”***

Add the %CSP.Page Super Class to the generated Classes (“X” must also be specified). The OnPage() method will be overridden to display the output of the XMLDump() method.

* 1. ***XML - “X”***

Add the %XML.Adaptor Super Class to each of the generated Classes. An XMLDump () method will be created to export the entire file in XML format. This uses the inherited XMLExport () instance method.

* 1. ***PACKAGE (Optional)***

The Caché Package for generated Classes. If not passed in, the default Package “VISTA” will be used, The Package name cannot be a SQL Reserved Word.

* 1. ***“ID” - (Optional)***

This value overrides the default ID string of “IEN” for each Class, if Simple IDs are requested by passing in the “I” flag.

* 1. ***OWNER (Optional)***

A valid Caché SQL Username. The classes will be created with this user as the owner. Although this flag is technically optional, it should be passed with a value of “HDSHDR”. The HDSHDR user has the necessary permissions to access a process necessary Classes, Methods and Tables.

Currently the call from - (CASH.HDR.Utility).VISTA() is:

**D ##CLASS(CASH.Utility).All("","DHhILNQ"\_$S($G(XREFS):"",1:"x"),"VISTA","RowId","HDSHDR")**

Note “HDSHDR” passed as the last parameter.

* 1. ***LIST (Optional)***

Array of fields to include in the mapped Class(es). All other Fields will be ignored if this array is passed. NOTE: If LIST is not passed, all fields will be mapped The LIST should be in the format:

LIST(**file#**,**field#1**)=""

…

LIST(**file#**,**field#n**)=""

Only the specified fields will be added to the Class (plus any required keys).

To specify fields in Multiples or Pointers, use the relevant Flags (“M” or “P”) and add array nodes as follows:

LIST(**multiple\_file#**,**field#1**)=""

...

LIST(**multiple\_file#**,**field#n**)=""

LIST(**pointer\_file#**,**field#1**)=""

...

LIST(**pointer\_file#**,**field#n**)=""

Example:

LIST(4,.01)="" - .01 field (Name) for file# 4 (Institution)

LIST(4,.02)="" - .02 field (State) pointer to file #5

LIST(5,.01)="" - .01 field (Name) for file# 5 (State)

LIST(5,3)="" - 3 field (County) multiple file# 5.01

LIST(5.01,.01)="" - .01 field (County) for file #5.01

This produces 3 Classes, assuming the “M” and “P” Flags are passed, each with one data Property, plus the relevant Relationships and Foreign Keys. If the “E” Flag is also passed a Computed Field will also be generated for the Pointer.