| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - cell.cpp (source / [functions](http://docs.google.com/cell.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 41 | 50 | 82.0 % | | Date: | 2012-03-23 |  | Functions: | 6 | 6 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 6 | 10 | 60.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   2 : : \* Copyrighted 2007 by Interra Systems, USA  3 : : \* All rights reserved   4 : : \*  5 : : \* Date of Creation: Fri Jul 20 15:52:01 IST 2007  6 : : \* File Name: cell.cpp  7 : : \* Purpose:   8 : : \* Author:   9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   12 : :   13 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  14 : : \* Header Include section   15 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  16 : : #include <stdio.h>  17 : : #include "cell.h"  18 : : #include "link.h"  19 : : #include "Porting.h"  20 : : #include "ErrorHandler.h"  21 : : #include "Logger.h"  22 : : #include "PH4Flag.h"  23 : : #include "CompressionFlag.h"  24 : :   25 : : using namespace std;  26 : :   27 : : namespace nft {  28 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  29 : : \* Constructor: VSBCellReader()  30 : : \* Purpose: To construct a VSBCellReader  31 : : \* Parameters: char\* pCFile,  32 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  33 : :   34 : : VSBCellReader::VSBCellReader(char\* pCFile   35 : : )  36 : 2188 : {  37 : 1094 : cellReInit(pCFile);  38 : : }  39 : :   40 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  41 : : \* Function Name: VSBCellReader()  42 : : \* Purpose: Initialize the cell file for reading from the buffer.  43 : : \* Parameters: char\* pCFile  44 : : \* Return Values: void  45 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  46 : :   47 : : void VSBCellReader::cellReInit (char\* pCFile)  48 : 1094 : {  49 : 1094 : MagicNumber magicNum ; //PH4 handling  50 : 1094 : Byte \*tmpMagic;  51 : :   52 : 1094 : mCellFile = pCFile;   53 : : //SegmentHeaderSize(sizeof(vsbPatternSegmentHeader))  54 : : {  55 : : //Now decode the header of the cell.n file  56 : 1094 : int \*length;  57 : 1094 : int l\_length[4] = {0, 0, 0, 0}; //length specific to LINUX  58 : : //This was l\_length[5] in E5K  59 : : //#ifdef ERROR\_SEED //<<Test\_Id:cell\_mCellFile\_Set\_To\_NULL>>  60 : : // mCellFile = NULL;  61 : : //#endif  62 [ - + ]: 1094 : if (NULL == mCellFile)  63 : : {  64 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  65 : : "VSB Cell Data is (null)");  66 : :   67 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  68 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  69 : : ,ErrorHandler::eErrMessageString  70 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  71 : : ,"VSB Cell data is NULL"));  72 : : }  73 : 1094 : cellPatSegment\_[VSB\_PAT\_HEADER] = pCFile;  74 : 1094 : length = (int\*)cellPatSegment\_[VSB\_PAT\_HEADER];  75 [ + + ]: 4376 : for (int i=0; i<NUM\_CELL\_SEGMENTS; i++)  76 : : {  77 : 4376 : l\_length[i] = BE2LE\_32(length[i]);  78 : : }  79 : 1094 : length = l\_length;  80 : : //Fetch the segment lengths   81 : 1094 : cellPatSegLength\_[VSB\_PAT\_HEADER] =   82 : : (length[1] \* WordSize) ;  83 : 1094 : cellPatSegLength\_[VSB\_PAT\_SEG\_POINTER] =   84 : : (length[2] \* WordSize) ;   85 : 1094 : cellPatSegLength\_[VSB\_PAT\_EXTENSION] =   86 : : (length[3] \* WordSize) ;   87 : :   88 : : //Get the segment offsets  89 : : //cellPatSegment[VSB\_PAT\_HEADER] is   90 : : // already populated above  91 : 1094 : cellPatSegment\_[VSB\_PAT\_SEG\_POINTER] =  92 : : cellPatSegment\_[VSB\_PAT\_HEADER]   93 : : + cellPatSegLength\_[VSB\_PAT\_HEADER];  94 : 1094 : cellPatSegment\_[VSB\_PAT\_EXTENSION] =  95 : : cellPatSegment\_[VSB\_PAT\_SEG\_POINTER]  96 : : + cellPatSegLength\_[VSB\_PAT\_SEG\_POINTER];  97 : 1094 : cellPatSegment\_[VSB\_PAT\_DATA\_COMPOUND] =  98 : : cellPatSegment\_[VSB\_PAT\_EXTENSION]  99 : : + cellPatSegLength\_[VSB\_PAT\_EXTENSION];  100 : :   101 : 1094 : tmpMagic = cellPatSegment\_[VSB\_PAT\_HEADER] ;  102 [ + - ]: 1094 : if(tmpMagic !=NULL)  103 : 1094 : memcpy(&magicNum,tmpMagic,4);  104 : :   105 : 1094 : uint8\_t mask = 0x01;  106 : 1094 : uint8\_t comp\_mask = 2;  107 : 1094 : PH4Flag::instance()->setValue( (magicNum.flag & mask)?1:0 );  108 : 1094 : CompressionFlag::instance()->setValue( (magicNum.flag & comp\_mask)?1:0 );  109 : : }  110 : : }  111 : :   112 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  113 : : \* Function Name: cellGetPatternAddress()  114 : : \* Purpose: Returns the pattern address for the corresponding  115 : : \* relation record.  116 : : \* Parameters: const vsbRelationRecord \*rr  117 : : \* Return Values: char\*  118 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  119 : : char\* VSBCellReader::cellGetPatternAddress(const vsbRelationRecord \*rr) const  120 : 1261 : {  121 : 1261 : int64\_t segOffset = 0;  122 : : #ifndef SOLARIS //default linux  123 : 1261 : int16\_t l\_patSegIndex = BE2LE\_16((int16\_t)rr->PatSegmentIndex);  124 : : //#ifdef ERROR\_SEED //<<Test\_Id:cell\_l\_patSegIndex\_Set\_To\_0>>  125 : : // l\_patSegIndex = 0;  126 : : //#endif  127 [ - + ]: 1261 : if ( l\_patSegIndex < 0) //Null pattern data  128 : : {  129 : 0 : return NULL;  130 : : }  131 : : else  132 : : {  133 : : //#ifdef ERROR\_SEED //<<Test\_Id:cell\_l\_patSegIndex\_Set\_To\_0>>  134 : : // uint16\_t l\_pat\_add = static\_cast<uint16\_t>(cellPatSegLength\_[VSB\_PAT\_SEG\_POINTER]);  135 : : // l\_pat\_add = 0;  136 [ - + ]: 1261 : if ( static\_cast<uint16\_t>(l\_patSegIndex + 1)   137 : : \* SegmentIndexSize > static\_cast<uint16\_t>(cellPatSegLength\_[VSB\_PAT\_SEG\_POINTER]) )  138 : : {  139 : : //#endif  140 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  141 : : "Invalid Segment Index");  142 : :   143 : 0 : char buf[BUFSIZ];  144 : 0 : sprintf(buf, "Invalid Segment Index %d\n", l\_patSegIndex);  145 : :   146 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  147 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  148 : : ,ErrorHandler::eErrMessageString  149 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  150 : : ,buf));  151 : : }   152 : : else  153 : : {  154 : : //Normal flow  155 : : }  156 : : }   157 : 1261 : segOffset = BE2LE\_64  158 : : (\*(uint64\_t \*)  159 : : (cellPatSegment\_[VSB\_PAT\_SEG\_POINTER]  160 : : + l\_patSegIndex \*   161 : : (SegmentIndexSize)  162 : : ));  163 : 1261 : return ( cellPatSegment\_[VSB\_PAT\_DATA\_COMPOUND]  164 : : + segOffset \* WordSize  165 : 1261 : + BE2LE\_32(rr->PatDataPointer) \* WordSize  166 : : + SegmentHeaderSize  167 : : );  168 : :   169 : : #else  170 : : //#ifdef ERROR\_SEED //<<Test\_Id:cell\_rr\_PatSegmentIndex\_Set\_to\_0>>  171 : : // rr->PatSegmentIndex = 0;  172 : : //#endif  173 : : if (rr->PatSegmentIndex < 0)  174 : : {  175 : : return NULL;  176 : : }  177 : : else  178 : : {  179 : : //#ifdef ERROR\_SEED //<<Test\_Id:cell\_rr\_PatSegmentIndex\_Set\_to\_0>>  180 : : // uint32\_t rr\_pat\_add = ( (uint16\_t)rr->PatSegmentIndex + 1) \* SegmentIndexSize;  181 : : // rr\_pat\_add = 65536;  182 : : if ( ( (uint16\_t)rr->PatSegmentIndex + 1) \* SegmentIndexSize >  183 : : cellPatSegLength\_[VSB\_PAT\_SEG\_POINTER])  184 : : {  185 : : //#endif  186 : : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  187 : : "Invalid Segment Index");  188 : :   189 : : ErrorHandler \*pErr = ErrorHandler::instance();  190 : : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  191 : : ,ErrorHandler::eErrMessageString  192 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  193 : : ,"Invalid Segment Index"));  194 : : }  195 : : else  196 : : {  197 : : //Normal flow  198 : : }   199 : :   200 : : }  201 : :   202 : : segOffset = \*(uint64\_t \*)  203 : : (cellPatSegment\_[VSB\_PAT\_SEG\_POINTER]  204 : : + rr->PatSegmentIndex   205 : : \* SegmentIndexSize);  206 : : return ( cellPatSegment\_[VSB\_PAT\_DATA\_COMPOUND]  207 : : + segOffset \* WordSize  208 : : + rr->PatDataPointer \* WordSize  209 : : + SegmentHeaderSize   210 : : );  211 : : #endif  212 : : }   213 : :   214 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  215 : : \* Function Name: cellGetPatternSegmentHeader()  216 : : \* Purpose: Retrieves the Pattern Segment Header of a cell  217 : : \* Parameters: const vsbRelationRecord \*rr,  218 : : \* vsbPatternSegmentHeader \*\*SegmentHeader)  219 : : \* Return Values: void  220 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  221 : : void VSBCellReader::cellGetPatternSegmentHeader(  222 : : const vsbRelationRecord \*rr,  223 : : vsbPatternSegmentHeader \*\*SegmentHeader) const  224 : 1261 : {  225 : 1261 : int64\_t SegmentOffset = 0 ;  226 : 1261 : SegmentOffset = BE2LE\_64(\*(uint64\_t \*)  227 : : (cellPatSegment\_[VSB\_PAT\_SEG\_POINTER]  228 : : + BE2LE\_16((short int)rr->PatSegmentIndex)   229 : : \* SegmentIndexSize  230 : : )  231 : : ) ;  232 : 1261 : \*SegmentHeader = (vsbPatternSegmentHeader \*)  233 : : (cellPatSegment\_[VSB\_PAT\_DATA\_COMPOUND]   234 : : + SegmentOffset \* WordSize) ;  235 : : }  236 : :   237 : 1215 : } //namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |