NISTIR 5762

Standard Generalized Markup Language Test Suite Evaluation Report

Craig S. Russell

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Office Systems Engineering Group
Gaithersburg, MD 20899

NIST

NISTIR 5762

Standard Generalized Markup Language Test Suite Evaluation Report

Craig S. Russell

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Office Systems Engineering Group
Gaithersburg, MD 20899

November 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

SGML
TEST SUITE
EVALUATION REPORT

by

Craig S. Russell

Systems and Software Technology Division
Computer Systems Laboratory
National Institute of Standards and Technology

January 10, 1995

FY93 CALS TASK 9.2. "Evaluate SGML Test Suites."

Deliverables Report of Evaluation and Recommendations

- ABSTRACT -

NIST has been tasked by the Continuous Acquisition and Life-cycle Support (CALS) Project Office to organize an SGML Conformance Testing Service. The first step in producing a conformance testing service was to produce a thorough test suite. The goal of this project was to evaluate existing test suites for use as the basis within an SGML conformance testing service. This document describes the methodology used in analyzing an SGML Test Suite and the different types of tests run with the various SGML products at NIST's disposal.

Table of Contents

1 EXECUTIVE SUMMARY

NIST has been tasked by the Computer Acquisition and Life-Cycle Support Office (CALS) to organize a Standard Generalized Markup Language (SGML) Conformance Testing Service. The first step in producing a conformance testing service was to produce a thoroughly comprehensive Test Suite. The Exoterica Corporation's SGML Test Suite has been analyzed and determined to be the most thorough and complete SGML Test Suite available. Therefore, NIST has decided to use Exoterica's Test Suite for the SGML conformance testing service.

2 GOAL

The primary purpose of this project was to evaluate existing test suites and to qualify them for use as a basis for implementation within an SGML conformance testing service. Conformance testing of document interchange standards is a complex task. SGML has its own set of particular difficulties and implications. NIST developed a limited Test Suite for its parser which is based on CALS requirements. However, since the initial development of the NIST conformance Test Suite, International Organization for Standardization (ISO) 8879 (which covers SGML) was revised with the addition of Amendment 1. In addition, CALS requirements were also modified. Therefore, a complete evaluation of the existing SGML Test Suite was necessary. Analytical evaluation aided in producing a more coherent testing policy and test procedure report for the SGML conformance testing service.

Although currently there are no affordably priced products available to the general public off the shelf, the number of commercially available products utilizing SGML is increasing. However, to ensure consistency within parser development, it would be advantageous for users of SGML products, both in government and commercial arenas, to have an SGML conformance testing service.

3 BACKGROUND

Two (CBD) announcements were released in September 1992, requesting existing executable Test Suites or toolkits to support conformance testing of ISO 8879 and MIL-M-28001B. NIST received seven responses to the CBD announcements.

None of the vendors adequately addressed the CBD announcement. Most of these vendors offered no tools or Test Suites. The Office Systems Engineering (OSE) Group contacted various vendors looking for additional Test Suites/tools. After contacting numerous individuals both in this country and abroad, we identified the existence of three additional Test Suites besides the NIST Test Suite.

All three companies were sent letters under an official NIST letterhead to encourage them to submit their Test Suites. In March 1993, Exoterica Corporation submitted its Test Suite. The other two companies would not respond to our attempts to encourage them to submit their

Test Suites.

During development of the NIST parser, NIST designed its own in-house Test Suite. The NIST parser, based on MIL-M-28001 requirements, was never intended to be a complete SGML commercially available parser product.

4 METHODOLOGY

The following information is the methodology used for evaluating the Exoterica Test Suite. The beginning phase of the research commenced by our gaining an understanding of the Exoterica conformance Test Suite. ISO 8879 and the Exoterica (three ring notebook) version of the Test Suite were placed side by side. The two resources were scanned visually, and each Test module that covered an appropriate SGML production was identified. The SGML abstract syntax is specified by formal syntax productions. A production consists of a reference number, the name of the syntactic variable being defined, an equals sign, and an expression that constitutes the definition. Some test scripts from the Exoterica Test Suite provided coverage for only one segment of a production, while other scripts often overlapped several productions. The test scripts were manually read and analyzed to ensure that the three ring notebook version of the Test Suite did not contain hidden sometimes difficult to locate, publication type errors. The three ring notebook of the Test Suite allows for human review of the test scripts, as well as showing the parser results. When a selected parser was tested against a production script, the parser's output was expected to produce similar output specifications as those defined in the Test Suite.

5 CONCLUSION

Exoterica's Test Suite and the NIST Test Suite were analyzed to determine their completeness for use as a possible basis for the SGML Conformance Test Suite. It was acknowledged beforehand that the NIST Test Suite was not complete. The acquisition of the Exoterica Test Suite provided NIST a complete Test Suite to examine. Therefore, the majority of the evaluation time was spent examining the Exoterica Test Suite. The Exoterica Test Suite consists of over two thousand scripts and is a thorough SGML Test Suite. Approximately two-thirds of the Exoterica tests measure conformance to FIPS 152. Several months of extensive investigative testing was done on the Exoterica Test Suite and many controlled test runs were conducted with the Exoterica Test Suite against the different SGML products that NIST had in its possession. The research findings demonstrated that each vendor's product parsed a similar test script differently.

APPENDIX A

EVALUATION PROCEDURES

NIST obtained several SGML products to evaluate. The evaluation closely monitored a product's ability to detect, warn, and accurately report a document's status. Before conducting actual tests, research team members examined the product's features, attributes, and operating procedures. The criteria established for testing a product was to determine consistency and clarity of parser output. For example, script segments extracted from the Test Suite generated a batch process that tested a particular SGML production. These segments executed imbedded instructions to parse and compare product results. The batch process caused Product "A" to parse the test script. Product "B" parsed the same test script file after minor adjustments (parser name, etc.) to the batch file. The parsed output obtained from products "A" and "B" showed a wide range of results.

The following section discusses the procedures used while examining the Exoterica Test Suite. The NIST Test Suite and the Exoterica Test Suite were evaluated to determine if the two Test Suite's would allow for further development of a single formal conformance Test Suite. Initially the composition of the Exoterica Test Suite was unknown by NIST. The Exoterica Test Suite required a substantial amount of time to examine and thoroughly investigate its roughly twenty-three hundred scripts. Therefore, most of the research time was spent examining the Exoterica Test Suite.

The analyst examined the Test Suite and the SGML Standard side-by-side, looking at each language production and how the test scripts tested these productions. The Exoterica test Suite clearly identified the relative productions indicated in the ISO Standard 8879. Amendment 1 of ISO 8879 is included in the Exoterica Test Suite. A yellow marker pen was used to highlight each production or specific portion of a production that a test script covered. Some scripts tested only small portions of a production while others tested most or all of a production's properties. The entire Test Suite took several weeks to completely examine using this method.

The Test Suite's structure showed each test script and its corresponding Reference Application for SGML Testing (RAST) output. The format provided the analyst a visual illustration of each production and test. At the conclusion of a parse, the data was collected and compared against previously defined RAST output of the same production. This process helped to promote a more accurate test. In the course of evaluating the content of the Exoterica Test Suite, NIST conducted parsing tests using these modules against several SGML based products.

Before testing each product in NIST's possession, the following examination procedures were performed. The first requirement was to evaluate the product's features. This increased the

analyst's understanding of each product. Each SGML product was installed onto a single computer hard disk. This was done as prescribed by the basic installation instructions packaged with each product. Product manuals and "readme" files provided with the applications (especially sections covering parsing) were read. Another requirement consisted of conducting controlled test runs to understand firsthand the potential pitfalls which may later be encountered during formal testing. These actions helped to gage the possible performance characteristics during an actual test. This knowledge is vital to the researcher if later he/she is asked to provide further guidance to government or commercial officials wishing to increase their understanding of how the products performed. The information obtained from parser conformance testing has enabled NIST too better contribute to standards development committees. Having a firsthand, practical knowledge of systems is invaluable in designing new standards and establishing conformance testing policies.

This section outlines procedures maintained during testing. Batch files were written to automate the running of the Exoterica Test Suite modules. These batch files allowed each product's test results to be directed and handled in a prescribed manner. From the operating system prompt, command line statements using Exoterica query language constructs were issued as arguments which collected production tests modules.

The query function caused execution of a single executable batch file to be created. When this batch file was run, it grouped production tests along with corresponding RAST results into one output file. The Exoterica Test Suite executing from a CD-ROM contained over two thousand production scripts that are modularized into groups that test a given production. Test results were written to an external file in a predetermined format so that they could later be examined. To check the accuracy and flow of data prior to the actual test, portions of a Test Suite module were modified in size. The average module contained over fifty lines of instructions. The analyst reduced the file size to approximately five lines of instructions. Since each product produced application specific output formats that differed in description and structure from each parser, the test results were tweaked to produce a standard, structured output that allowed for easier follow-up evaluation. *Awk* scripts (data transformation, report generation language) and *DOS* batch files that the analysist developed were then used to inspect and organize parsed output results into alternate storage files. The scripts/batch files wrote to and extracted information from the test results and categorized the output as either a success, failure, warning, or other possible SGML violation. Other problems that may have occurred during the parse were also maintained in an external file. Initially, the Exoterica Test Suite scripts were used to confirm only test output that reported as being "Successful" (this means the parsing process correctly found and reported errors existing in a document or validated that file as having no errors). If the modified script module performed correctly, the entire module was run against the product. At the conclusion of each test, a single external output file was generated that contained the product's parsed data. The output file was stored electronically, printed, and pasted into a lab notebook.

AWK scripts were written to extract specific items of information from each output file following a completed test run. The information extracted by the *AWK* script was written to another file where it was stored, printed, and examined manually using a text editor. Manual examination of an output file was tedious and overwhelming, because there were over one thousand results stored in that file. A query to inspect for minimal SGML conformance (conforming to FIPS 152) produced a batch file containing over fourteen hundred test scripts. At the completion of the test run, an output file was produced that contained over fourteen hundred results. Therefore, it became apparent that the evaluation of the output files had to be automated. The manual examination process was automated by the development of script files that looked for certain streams of data. Scripts varied with each completed test result file. The evaluation process culminated with the cataloging and archiving of test result data.

The goal of this entire process was to determine if the parsed output produced by a product was accurate, reliable and consistent. The evaluation shifted its focus to those production modules known to produce only errors when parsed. If the parser "passed" a file as having no errors when in fact it did have errors, the parser would indicate an inability with the product to recognize an invalid file. It should be noted here that some test script modules that did not produce errors were altered so that they would produce an error when parsed. This was done as an additional check on the parser. Test scripts were designed to check for valid and invalid production states. Therefore, *Awk* scripts were written to find lines in an output file that had the words "unsuccessful" or "error" and place these lines and file names into a separate report file.

Again, only those lines of text in an output file indicating an error were checked. For each module reporting an error the analyst examined that test module along with the corresponding source file that was parsed to determine the validity of this finding. Whenever a parse error was encountered in an output file the nature and cause of this error was determined. Was the test incorrectly run or did the product reach this false conclusion based on the parser's construction? This question was investigated thoroughly since parser construction is based on interpretations of the International Organization of Standandization (ISO). If the parser was producing incorrect results (validating an invalid source file), the analyst next sought to discover if the parser was interpreting ISO 8879 incorrectly or did the product simply lack coverage of some minimal SGML productions.

The inconsistencies and various output formats associated with parser output made testing SGML related products a challenging task. There are ambiguities within ISO Standard 8879 that allow for varied interpretations. Such ambiguities noted after the release of ISO 8879 and subsequent changes made to correct them with Amendment 1 may help lead to the development of parser consistency and accuracy.

APPENDIX B

SAMPLE TEST RESULTS

This appendix contains listings of a cross section of the tests that were performed. Hundreds of tests were performed in order to gain a working knowledge of the SGML products in NIST's possession. The primary researcher on this project kept a laboratory notebook which was used to document all tests. It contains the date and time of all tests and is maintained in a chronological order. It features a written summation of each experiment conducted, along with a description of the source file, parser, date, test number, and result of each experiment. The number of experiments performed filled multiple notebooks.

Many tests were performed in order to gain the knowledge necessary to develop a structured, official testing methodology. Each of the products offered semantic-specific functionality. Each of the products required the input document to be in a particular structure and produced parsed output in an encoded, system-defined form. The customer using a SGML product must be confident of the parsing accuracy in their product and also receive clear and non-evasive output comments. Only one product produced parser output in a RAST-like format. The RAST structure is the official ANSI format specifying how an SGML parser should define output. One of the products examined in the test required that the SGML declaration be altered due to the semantics of the document manager. While interpretation of the ISO 8879 is challenging, the conventions needed to utilize SGML-related products must be consistent to ensure correct, meaningful and understandable results. What is meant by compliant or non-compliant for a given document must be accurately stated. Some of the products allow the user to supply their own SGML declaration while other products only permit the use of a default declaration.

The following pages contain samples of the test runs that were performed. They are displayed to give the reader some idea of the various types of tests that were conducted. In no way are these listings meant to show favoritism or fault toward a particular vendor's product as this was not the goal of this project. The excerpts that follow were taken from the lab notebook maintained during the evaluation, and the experiments should be self-explanatory.

Experiment Summary

DATE 03-26-93

Summary: The source file **dtdsampl.txt** was moved from the DOS (plain ASCII text) format and converted to UNIX format via the dos2unix command utility. The file was transferred to an UNIX based operating system using file transfer protocol. The source files were stored in directory "~/wptestsuite." The source file dtdsample.txt was checked for errors which may have occurred during file transfer. And all "cap M" characters (^M) encountered were removed before parsing using the VI editor. The com-mand line argument issued for the test was: "dtd2lgc dtdsampl.txt -s sgml." In this test the default SGML declaration provided by the product was used.

This test does not make a value judgement for or against the use of the product or features. The test was conducted to determine the strength of output characteristics and not product capabilities.

Test Descriptions

Test Number: **TR3-26.01**

Test description
Source file: **dtdsampl.txt**

Results: Successful Completion

Date: **3-26-93**
Source file: **dtdsampl.txt**
Test number: **TR3-26.01**

Command line: **dtd2lgc dtdsample -s sgml**

Processed on: SUN UNIX system Desktop SPARC

Result of parse :

External Entity Mapping File = None
SGML Declaration File =
/home/russell/wptestsuite/sgml

(Line 43, dtdsample)
Warning: Parse completed before end of file

*** SUCCESSFUL COMPLETION! ***

Experiment Summary

DATE: 3-31-93

Summary: The source file's name was **mdtdpro.unx.** This was done to test a wider variety of file structures against the parser. By attaching the preceding letter "m" to dtdpro.unx, the analyst could later track the origin of the file. The "m" stands for Macintosh, the system from which the document was obtained. The file was converted from plain ASCII text to UNIX format. The file was then parsed. The command line argument was: "dtd2lgc mdtdpro.unx". The source file was stored in dir: "~/wptestsuite." The source file was checked for errors. No changes were made to the file and it was parsed. It is only the resulting output that is of interest here.

Test Descriptions

Test Number: **TR3-31.01**

Test description
Source file: **mdtdpro.unx**

Results: "Expecting sgml or Doctype after mdo" was the reported output. The error was detected successfully by the parser. The file's Doctype statement is public, but all the appropriate links to the associated file were not supplied.

Date: 3-31-93
Source file: **mdtdpro.unx**
Test number: **TR3-31.01**

Command line: **dtd2lgc mdtdpro.unx -s sgml**
Processed on: Sun UNIX system Desktop SPARC

Result of parse:

External Entity Mapping File = None
SGML Declaration File = /home/russell/wptestsuite/sgml

(Line 13, mdtdpro.unx)
SGML error:
Expecting SGML or DOCTYPE after MDO delimiter ('<!')

***** LGC file not written! *****

Experiment Summary

DATE: 6-22-93

Summary: The source file, **compltdsample,** is a complete SGML document. Dtd and document instance were intact and parsed. The file contained the example text used for the Output Specification of MIL-M-28001A. The source file was checked for errors in text using the VI text editor. The command line argument issued for the test was: "dtd2lgc compltdsample -s sgml." The application's default declaration was used for this test, as opposed to the declaration supplied with the actual file. The output produced by the parser was analyzed; not the product's functionality.

There were no errors found in the source document before it was parsed.

Test Descriptions

Test Number: **TR6-22.01**

Test description
Source file: **compltdsample**

Results: Successful Completion.
The source file was given the .lgc extension after a successful parse.

Date: 6-22-93
Source file: **compltsample**
Test number: **TR6-22.01**

Command line: **dtd2lgc compltsample -s sgml**

Processed on: Sun System Desktop SPARC

Result of parse :

External Entity Mapping File = None
SGML Declaration File = /home/russell/wptestsuite/sgml

(Line 43, compltsample)
Warning: Parse completed before end of file

*** SUCCESSFUL COMPLETION! ***

Experiment Summary

DATE: 6-22-93

Summary: Parsed source file named **compltsa**. The file **compltdsample** used in earlier tests is the same document marked for this test. The file name was shortened as a result of the name length restriction of eight characters for a file stored on a MS-DOS operating system. The original UNIX format of the file was transferred via ftp to a PC desktop computer. See the lab book for a complete parser list and further information on the origin of this product. The source file was checked for textual errors using a VI editor. The command line argument was: "SGMLS compltsa." The default SGML declaration was used in this test as well.

Test Descriptions

Test Number: **TR6-22.02**

Test description
Source file: **compltsa**

Results: "Successful Completion." The parser also pointed out warnings regarding external general entities it could not find. The parser is correct in issuing these warnings. There were no mapping files supplied for these external entities. Note the difference in the resulting structures of file output classifications.

Date: 6-22-93
Source file: **compltsa**
Test number: **TR6-22.02**

Command line: **SGMLS compltsa**
Processed on: Unitek 486

Result of parse :

```
sgmls: Error at compltsa, line 7 in declaration parameter 5:
               Could not find external general entity "f15eagle"
sgmls: Error at compltsa, line 9 in declaration parameter 5:
               Could not find external general entity "apcard"
TOTALCAP   995/35000
ELEMCAP    192/35000
GRPCAP     480/35000
EXGRPCAP    16/35000
EXNMCAP     16/35000
ATTCAP      56/35000
ATTCHCAP     8/35000
AVGRPCAP   112/35000
NOTCAP      16/35000
NOTCHCAP    99/35000
```

Experiment Summary

DATE: 6-22-93

Summary: Parsed source file named **cals.doc**. The file's original name was retained as cals.doc. The file is a complete conforming SGML document. No errors were found in the file prior to its being parsed. The command line for the test run was: "SGMLS cals.doc."

Test Descriptions

Test Number: **TR6-22.07**

Test description
Source file: **cals.doc**

Results: "Successful Completion." The parser also pointed out additional warnings made regarding external entities it could not locate on the system. The parser warning messages are appropriately specified. There were no mapping links establishing the location of external entities.

Date: 6-22-93
Source file: **cals.doc**
Test number: **TR6-22.07**

Command line: **SGMLS cals.doc**
Processed on: Unitek 486

Results of parse:

```
sgmls: Warning at cals.doc, line 20 in declaration parameter 28:
        Unrecognized designating escape sequence "ESC 2/13 4/3"
sgmls: Error at cals.doc, line 477 in declaration parameter 5:
        Could not find external parameter entity "ISOlat1"
sgmls: Error at cals.doc, line 480 in declaration parameter 5:
        Could not find external parameter entity "ISOpub"
sgmls: Error at cals.doc, line 483 in declaration parameter 5:
        Could not find external parameter entity "ISOgrk3"
sgmls: Error at cals.doc, line 486 in declaration parameter 5:
        Could not find external parameter entity "ISOnum"
sgmls: Error at cals.doc, line 489 in declaration parameter 5:
        Could not find external parameter entity "ISOtech"
TOTALCAP 118330/175000
ENTCAP  2880/35000
ENTCHCAP  4857/35000
ELEMCAP  4736/35000
GRPCAP 49760/70000
EXGRPCAP   448/35000
EXNMCAP   864/35000
ATTCAP 37024/50000
ATTCHCAP   513/35000
AVGRPCAP 17216/35000
NOTCAP    32/35000
```

Experiment Summary

DATE: 6-22-93

The files text was corrupted to change valid output to invalid. The document was repeatedly executed through the parser to ensure accurate parsing output. The command line argument issued for the test is "SGMLS cals.doc."

Test Descriptions

Test Number: **TR6-22.08**

Test description
Source file: **cals.doc**

Results: "The file parsed successfully with one warning." The warning was given due to a reference to an unrecognized escape sequence in the SGML Declaration. The warning did detect the change made to the file.

Date: **6-22-93**
Source file: **cals.doc**
Test number: **TR6-22.08**

Command line: **SGMLS cals.doc**

Processed on: Unitek 486

Result of parse:

sgmls: Warning at cals.doc, line 20 in declaration parameter 28:
Unrecognized designating escape sequence "ESC 2/13 4/3"
TOTALCAP 118330/175000
ENTCAP 2880/35000
ENTCHCAP 4857/35000
ELEMCAP 4736/35000
GRPCAP 49760/70000
EXGRPCAP 448/35000
EXNMCAP 864/35000
ATTCAP 37024/50000
ATTCHCAP 513/35000
AVGRPCAP 17216/35000
NOTCAP 32/35000

Experiment Summary

DATE: 6-22-93

Summary: Parsed source file **cals.doc**. The file name was changed to cals.txt. The file was sent via ftp to an UNIX system and placed in a directory called "~/workinhere." This directory was used to store all files and results while testing this parser. Before testing, the source file was checked for textual errors and all "cap M's" (^M), were removed using the VI editor extract feature. The application's GUI was used to conduct all tests.

Test Descriptions

Test Number: **TR6-22.10**

Test description
Source file: **cals.doc**

Results: Successful completion, but there were four warnings given. All warning messages correspond to a lack of external entity mapping links. Their output file is richer in details.

Date: **6-22-93**
Source file: **cals.doc**
Test number: **TR6-22.10**

Command line: GUI driven options

Processed on: Sun UNIX system Desktop SPARC

Result of parse:

Dtgen Parser Warning:
While parsing file cals.txt-da/cals.txt/cals.txt.dtd:
Potential problem in the mixed content model of element CALLOUT. At some location within the model, the entry of separators between tags will not always be permitted. It is often possible to fix this problem by writing the #PCDATA model token in a repeatable OR group.
(err:768 line:1108 pos:43806)

Dtgen Parser Warning:
While parsing file cals.txt-da/cals.txt/cals.txt.dtd:
Potential problem in the mixed content model of element DEF. At some location within the model, the entry of separators between tags will not always be permitted. It was often possible to fix the problem by writing the #PCDATA model token in a repeatable OR group.
(err:768 line:1108 pos:43806)

Dtgen Parser Warning:
While parsing file cals.txt-da/cals.txt/cals.txt.dtd:
Potential problem in the mixed content model of element ENTRY. At some location within the model, the entry of separators between tags will not always be permitted. It is often possible to fix this problem by writing the #PCDATA model token in a repeatable OR group.
(err:768 line:1108 pos:43806)

Dtgen Parser Warning:
While parsing file cals.txt-da/cals.txt/cals.txt.dtd:
Potential problem in the mixed content model of element HOWTOUSE. At some location within the model, the entry of separators between tags will not always be permitted. It is often possible to fix this problem by writing the #PCDATA model token in a repeatable OR group.
(err:768 line:1108 pos:43806)

Dtgen Application Warning:
These Fosi variables are defined but not referenced:
dashfill (CHARFILL)

bodyfolio (ARABIC)
figct (ARABIC)
listct (ARABIC)
sectct (ARABIC)
wdotfill (CHARFILL)
tabct (ARABIC)
subpara2ct (ARABIC)
seqlistct (ARABIC)
subpara3ct (ARABIC)
foldoutct (ARABIC)
chapct (ARABIC)
graphicct (ARABIC)
subpara1ct (ARABIC)
paract (ARABIC)
frontfolio (ARABIC)
appfolio (ARABIC)
titlefolio (ARABIC)
subfigct (ARABIC)
spcfill (CHARFILL)
rearfolio (ARABIC)
dotfill (CHARFILL)

Experiment Summary

DATE: 8-11-93

Summary: The page below shows the batch files employed for automating the execution of this parser's test. The system's autoexec.bat file was changed to incorporate the new file directory. The batch file was put in the correct directory. Also, the special program code needed (scripting language), was placed in the same directory. The file containing the scripting language code was named "qmp.xom." The command line arguments are found in the file named omnisea.bat. The files parsed with this parser were stored in a separate subdirectory. The batch file was put in a directory named "omnitest," while the source files were placed in a subdirectory named "example."

Test Descriptions

Test Number: **TR8-11.01**

Test description:
Source file: Selected 6 Production test scripts from the vendor's package

All errors in the files were reported and one file passed as it was expected.

```
@echo off
REM The file name is omnisea.bat
REM and executes a for statement
REM file name is omnitsrc.bat
for %%1 in (example\*.*) do call omnisea.bat %%1
echo FINISHED
```

```
@echo off
REM file name is omnisea.bat
REM  runs parser output to a file
echo  ********************** >> TR8-11.01
echo  Parser Output >> TR8-11.01
echo  FILE %1 >> TR8-11.01
omnimark %1  -s qmp.xom -alog  TR8-11.01
echo  ********************** >> TR8-11.01
```

Date: **8-11-93**
Source file: **Selected 6 scripts**
Test number: **TR8-11.01**

Command line: see omnisea.bat on 8-11-93

Processed on: Unitek 486

Result of parse :
****
Parser Output
FILE example**GEPA3001**
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.
****
****
Parser Output
FILE example**P7579411**
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.

OMNIMARK.EXE --
SGML Error on line 43 in file example\P7579411:
An attribute other than a CDATA attribute must not have a null literal
("") value or a value that does not have any name tokens in it.
In the ATTLIST for element "P75-G1", the default value for the IDREF
attribute "P75-A2" was empty.

There was 1 SGML error detected.
****
****
Parser Output
FILE example**P7D79411**
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.

OMNIMARK.EXE --
SGML Error on line 41 in file example\P7D79411:
An attribute other than a CDATA attribute must not have a null literal
("") value or a value that does not have any name tokens in it.
In the ATTLIST for element "P7D79411", the default value for the NUMBERS
attribute "P7D-A1" was empty.

```
There was 1 SGML error detected.
************************
************************
Parser Output
FILE example\IFM94101
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.

OMNIMARK.EXE --
SGML Error on line 61 in file example\IFM94101:
The total number of opened entities must not exceed ENTLVL.
The entity being opened was "ifm-e17", and there are already 16 entities
opened (in addition to the document entity).

There was 1 SGML error detected.
************************
************************
Parser Output
FILE example\IF194303
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.

OMNIMARK.EXE --
SGML Error on line 47 in file example\IF194303:
An entity must not end in a processing instruction.
The entity was "if1-e1".

There was 1 SGML error detected.  OMNIMARK.EXE --
************************
************************
Parser Output
FILE example\IF394303
OmniMark V2R2
Copyright, (C) 1991 by Exoterica Corporation.

OMNIMARK.EXE --
SGML Error on line 45 in file example\IF394303:
An entity must not end inside a comment.
The entity was "if3-e1".
There was 1 SGML error detected.
```