

Extensible Markup Language (XML) 1.0

1. Εισαγωγικά

Σε ένα κόσμο όπου οι πληροφορίες παρέχονται μέσω του παγκόσμιου διαδικτύου, τα έγγραφα πρέπει να είναι εύκολα προσβάσιμα, μεταφέριμα και ευέλικτα. Πρέπει επίσης να είναι ανεξάρτητα οποιουδήποτε συστήματος και περιεχομένου. Οι γενικευμένες γλώσσες έχουν τέτοια χαρακτηριστικά, παρέχοντας στα έγγραφα αυτά μια δυνατότητα η οποία δεν υπάρχει σε άλλες γλώσσες περιγραφής εγγράφων. Η HTML είναι προβληματική και περιοριστική γλώσσα. Η XML έλυσε πολλά από τα προβλήματα που αντιμετώπισαν οι σχεδιαστές του web και είναι υπεύθυνη για την XHTML, μια ανασχεδιασμένη HTML. Θα χρησιμοποιείται για πολλά χρόνια επειδή προσφέρει αποτελεσματικές και δυναμικές πολυμεσικές λύσεις.

Η XML σχεδιάστηκε να ικανοποιήσει πολλές ανάγκες δίνοντας στα έγγραφα ένα μεγαλύτερο επίπεδο προσαρμοστικότητας στο στυλ και τη δομή από αυτό που υπήρχε παλαιότερα στην HTML. Η XML προσφέρει στους σχεδιαστές της HTML τη δυνατότητα να προσθέτουν περισσότερα στοιχεία στη γλώσσα. Δεν αναφέρεται μονάχα στους σχεδιαστές του web αλλά σε οποιονδήποτε ασχολείται με εκδόσεις.

Στην πραγματικότητα, η XML είναι markup γλώσσα για έγγραφα που περιέχουν δομημένες πληροφορίες.

Markup γλώσσα είναι ένας μηχανισμός που καθορίζει δομές σε ένα έγγραφο.

Οι δομημένες πληροφορίες περιλαμβάνουν περιεχόμενο και κάποιες διευκρινίσεις για το ρόλο που παίζει το περιεχόμενο. σχεδόν όλα τα έγγραφα έχουν την ίδια δομή. Η XML είναι κάτι περισσότερο από markup language είναι metalanguage, δηλαδή μια γλώσσα που χρησιμοποιείται για να καθορίσει νέες markup γλώσσες.

Η XML συμπληρώνει και δεν αντικαθιστά την HTML. Ενώ η HTML χρησιμοποιείται στη διατύπωση και την εμφάνιση των δεδομένων η XML αναπαριστά τη συναφή έννοια των δεδομένων. Στην HTML τα tags είναι προκαθορισμένα ενώ η XML παρέχει τη δυνατότητα να καθορίζουν οι χρήστες τα tags και τις δομημένες μεταξύ τους σχέσεις.

Τα XML έγγραφα δεν είναι πολύπλοκα αλλά απλά και πολύ αποτελεσματικά. Το διδακτικό υλικό της well-formed XML αναλύει τη δημιουργία των XML εγγράφων, η οποία είναι κατά κάποιο τρόπο ίδια με την HTML καθώς επιτρέπει τη μη δομημένη δημιουργία εγγράφου. Η valid XML είναι πιο σύνθετη. Απαιτεί την ύπαρξη ενός Document Type Definition πριν να γραφεί το έγγραφο αλλά παρέχει μια γενική δομή με βάση την οποία τη δημιουργούμε.

Η γλώσσα προγραμματισμού XML περιγράφει μια κατηγορία πληροφοριών (data objects) που καλούνται XML έγγραφα (documents) καθώς επίσης περιγράφει τμηματικά τη συμπεριφορά των προγραμμάτων που τα επεξεργάζονται.

Τα XML έγγραφα αποτελούνται από μονάδες αποθήκευσης που καλούνται entities (οντότητες), οι οποίες περιέχουν πληροφορίες αναλυμένες ή μη. Οι αναλυμένες πληροφορίες αποτελούνται από χαρακτήρες (characters) οι οποίοι συνθέτουν character data και άλλοι οι οποίοι συνθέτουν markup. Η μορφή markup κωδικοποιεί την περιγραφή της τελικής αποθήκευσης του εγγράφου καθώς και τη λογική δομή.

Ένα λογισμικό μοντέλο που καλείται **επεξεργαστής XML** χρησιμοποιείται να διαβάσει XML έγγραφα και παρέχει πρόσβαση στο περιεχόμενο και τη δομή τους. Υποτίθεται ότι ο επεξεργαστής XML λειτουργεί εκ μέρους ενός άλλου μοντέλου που καλείται **application** (εφαρμογή). Αυτή η προδιαγραφή περιγράφει την απαιτούμενη συμπεριφορά του επεξεργαστή και συγκεκριμένα πως θα πρέπει να διαβάσει τα XML δεδομένα και ποιες πληροφορίες πρέπει να παρέχει στην εφαρμογή.

1.1 Προέλευση και στόχοι (Origin and goals)

Η γλώσσα XML αναπτύχθηκε από μια Ομάδα Εργασίας της XML κάτω από την καλή κηδεμονία του διεθνούς οργανισμού World Wide Web Consortium (W3C) το 1996. Εδραιώθηκε από τον John Bosak της Sun Microsystems με την ενεργή συμμετοχή μιας XML Ομάδας Ειδικού Ενδιαφέροντος (που οργανώθηκε από τον οργανισμό W3C).

Οι προσχεδιασμένοι στόχοι της XML είναι:

1. Η XML πρέπει να είναι εύχρηστη στο Internet.
2. Η XML πρέπει να υποστηρίζει μεγάλη ποικιλία από εφαρμογές.
3. Η XML πρέπει να είναι συμβατή με την SGML.
4. Θα είναι εύκολο να γράφονται προγράμματα που επεξεργάζονται XML έγγραφα.
5. Ο αριθμός των προαιρετικών χαρακτηριστικών στην XML θα είναι όσο το δυνατόν πιο μικρός, ιδανικό επίπεδο το μηδέν.
6. Τα XML έγγραφα θα πρέπει να είναι ευανάγνωστα.
7. Ο σχεδιασμός XML θα πρέπει να προετοιμάζεται γρήγορα.
8. Ο σχεδιασμός XML θα πρέπει να είναι τυπικός και περιεκτικός.
9. Τα XML έγγραφα θα πρέπει να δημιουργούνται εύκολα.
10. Η περιεκτικότητα στον XML συμβολισμό είναι μικρής σημασίας.

2. Έγγραφα (Documents)

2.1 Τι είναι τα well-formed και τι τα valid έγγραφα

Βασικά, υπάρχουν δυο τύποι XML εγγράφων : τα well-formed και τα valid.

Ένα well-formed XML έγγραφο ακολουθεί τους γενικούς κανόνες σύνταξης της XML, οι οποίοι είναι πιο αυστηροί από αυτούς της HTML και της SGML. Οι χαρακτήρες δεδομένων της XML δεν μένουν ποτέ δίχως ένα markup τέλους οποιουδήποτε είδους , είτε end-tag όπως το ζεύγος <MYTAG></MYTAG>, είτε ένα empty element tag με το σύμβολο της καθέτου πριν το σύμβολο >, όπως <MYTAG/>. Το markup της XML ξεκινάει πάντοτε με το σύμβολο < ή με το σύμβολο &. Οι τύποι των στοιχείων και τα ονόματα των εισαγωγικών είναι case sensitive. Τα χαρακτηριστικά απαιτούν εισαγωγικά κ.α.

Τα valid XML έγγραφα ακολουθούν ένα συγκεκριμένο Document Type Definition(DTD). Ευθύνη των συγγραφέων και των εκδοτών είναι να επιβεβαιώνουν την εγκυρότητα των XML εγγράφων, ενώ οι ικανοί XML browsers χρειάζονται μόνον τον έλεγχο για καλή μορφοποίηση εάν θέλουν να διαβάσουν XML έγγραφα. Έτσι κάθε XML parser ελέγχει το έγγραφο για καλή μορφοποίηση και εγκυρότητα ενώ ο browser αναζητά μονάχα την καλή μορφοποίηση.

Αν ένα data object είναι well-formed είναι ένα XML έγγραφο. Ένα well-formed XML έγγραφο μπορεί να είναι valid εάν πλήρη κάποιους περιορισμούς.

Κάθε XML έγγραφο έχει μια λογική και μια φυσική δομή. Φυσικά, το κείμενο συνθέτεται από μονάδες που καλούνται οντότητες (entities). Η οντότητα μπορεί να αναφέρεται σε άλλες οντότητες για να προκαλέσει τον συνυπολογισμό τους στο έγγραφο. Το έγγραφο ξεκινάει από την «αφετηρία» ("root") ή από την οντότητα του εγγράφου (document entity). Λογικά, το έγγραφο αποτελείται από δηλώσεις, στοιχεία, σχόλια, αναφορές σε χαρακτήρες και οδηγίες εκτέλεσης , καθένα από τα οποία φαίνονται στο έγγραφο με σαφές markup.

2.1.1 Well-formed XML έγγραφα

Ακολουθώντας τα παρακάτω βήματα και παραδείγματα μπορείτε να κατανοήσετε την έννοια της well-formed XML πραγματοποιώντας έτσι τους στόχους αυτού του διδακτικού υλικού.

Τα well-formed έγγραφα είναι καλά τροποποιημένα επειδή δεν είναι απαραίτητο να δημιουργούνται σε ένα δομημένο περιβάλλον, έναντι ενός προκαθορισμένου συνόλου δομημένων εικόνων, αλλά να συμμορφώνονται μερικά με τους καλά τροποποιημένους περιορισμούς. Αυτοί οι περιορισμοί απαιτούν ότι τα στοιχεία που ονομάζονται content containers, φωλιάζουν ιδανικά μεταξύ τους και χρησιμοποιούν σωστά άλλη σύνταξη markup. Τα well-formed XML στοιχεία καθορίζονται από τη χρήση τους, και όχι από έναν αυστηρά δομημένο ορισμό, επιτρέποντας τους συγγραφείς να δημιουργούν στοιχεία σε ανταπόκριση με την ανάπτυξή τους. Αυτή η ελαστικότητα προσφέρει στους συγγραφείς μεγαλύτερο έλεγχο γύρω από την επεξεργασία και τον σχεδιασμό ενός εγγράφου από αυτά που υπήρχαν στα παραδοσιακά SGML περιβάλλοντα, στα οποία η δομή έπρεπε να καθορίζεται τυπικά σε ένα DTD προτού γραφεί οποιοδήποτε έγγραφο.

Τα well-formed XML ελευθερώνουν τους συγγραφείς του WEB από την προκαθορισμένη φύση της HTML, επιτρέποντας την φαντασία να υπερισχύει των περιορισμών. Η HTML είναι ένας προκαθορισμένος τύπος εγγράφου, που σημαίνει ότι δεν μπορεί να επεκταθεί ή να αλλαχθεί για να βελτιώσει την περιγραφική του δύναμη. Για παράδειγμα, οι συγγραφείς μπορούν να περιγράψουν έγγραφα στην XML χρησιμοποιώντας στοιχεία με δικές τους ονομασίες, όπως ESSAY, SECTION, PARAGRAPH, NOTE και IMPORTANT. Εφόσον συγγράψει το έγγραφο ο συγγραφέας μπορεί να αλλάξει ένα παράδειγμα του στοιχείου PARAGRAPH σε TAKENOTICE για να επισημάνει ότι το παράδειγμα αυτό διαφέρει από τα προηγούμενα και εξυπηρετεί έναν συγκριτικό σκοπό. Αυτό το επίπεδο ελαστικότητας είναι ενδεικτικό της well-formed XML και επιτρέπει στους συγγραφείς να περιγράφουν τα έγγραφα όπως αισθάνονται ότι ταιριάζουν. Οι συγγραφείς μπορούν να δημοσιεύσουν τα XML έγγραφα με XSL, CSS, ή DSSSL stylesheets, τα οποία παρέχουν στους WEB browsers και στα εργαλεία μετατροπής styling πληροφορίες για κάθε στοιχείο για να παρουσιάζουν τα έγγραφα ιδανικά.

Παρακάτω ακολουθούν τα βήματα στα οποία μπορείτε να βασιστείτε για τη σύνταξη well-formed XML εγγράφων.

ΒΗΜΑ 1: Πως γίνεται μια XML δήλωση

Ο σκοπός αυτού του βήματος είναι να δημιουργήσουμε μια XML δήλωση, η οποία να δηλώνει τη φύση των XML εγγράφων στους αναγνώστες των εγγράφων. Χωρίς αυτή τη δήλωση ο αναγνώστης μπορεί να αναγνωρίσει λανθασμένα το έγγραφο ως SGML, HTML, ή RTF προκαλώντας μεταφραστικά προβλήματα.

Κάθε έγγραφο XML πρέπει να χρησιμοποιεί μια XML δήλωση, για να δηλώνει τη φύση του στους αναγνώστες των XML εγγράφων. Οι επεξεργαστές κειμένου, οι browsers και οι επεξεργαστές εγγράφων χρησιμοποιούν τη δήλωση για να καθορίσουν πώς ένα έγγραφο μπορεί να επεξεργαστεί. Η δήλωση είναι πολύ σημαντική στα μεγάλα και σύνθετα έγγραφα αλλά μπορεί επίσης να χρησιμοποιηθεί σε μικρότερα και δοκιμαστικά έγγραφα. Η XML δήλωση περιλαμβάνει πληροφορίες για τη γλώσσα markup, για την παρουσία εξωτερικών δηλώσεων markup και για την κωδικοποίηση χαρακτήρων.

Δήλωση XML

Αυτές οι XML δηλώσεις χρησιμοποιούνται για ποικίλους τύπους XML εγγράφων. Οι πρώτες δυο χρησιμοποιούνται για να περιγράψουν τα well-formed και τα valid XML έγγραφα, αντίστοιχα. Η τρίτη δήλωση μπορεί να θεωρηθεί ως default δήλωση

XML, δηλώνοντας ότι είναι έγγραφο της έκδοσης 1.0, δεν χρησιμοποιεί εξωτερικές δηλώσεις markup και είναι κωδικοποιημένη σε μορφή UTF-8.

```
<?xml version="1.0" standalone="yes"?>
<?xml version="1.0" standalone="no"?>
<?xml version="1.0" standalone="no" encoding="UTF-8"?>
```

Συστατικά της δήλωσης XML

Όρος	Ερμηνεία
<?xml	Αρχίζει την δήλωση XML, ένα είδος οδηγίας για την επεξεργασία.
version	Η δήλωση version περιγράφει την έκδοση της XML που χρησιμοποιείται, που πρέπει να είναι ίση με "1.0", καθώς η XML 1.0 είναι η τρέχουσα και μοναδική έκδοση της XML.
standalone	Η Standalone δήλωση εγγράφου επιτρέπει στον συγγραφέα του εγγράφου να διευκρινίζει αν υπάρχουν εξωτερικές markup δηλώσεις. Αυτό το χαρακτηριστικό μπορεί να είναι ίσο με yes ή no και είναι συνήθως το πρώτο στα well-formed έγγραφα.
encoding	Η encoding δήλωση επιτρέπει στον συγγραφέα να καθορίσει τον χαρακτήρα κωδικοποίησης που θα χρησιμοποιηθεί. Αυτή η δήλωση πρέπει να χρησιμοποιείται μόνο από συγγραφείς που χρησιμοποιούν χαρακτήρα κωδικοποίησης εκτός του US-ASCII, τον πιο κοινό, ή τον UTF-8.
?>	Κλείνει την XML δήλωση.

ΒΗΜΑ 2: Δημιουργώντας ένα στοιχείο αφετηρίας.

Ο στόχος αυτού του βήματος είναι να δημιουργήσουμε ένα στοιχείο αφετηρίας, που να περιέχει όλα τα στοιχεία που δημιουργείτε μετέπειτα. Είναι το πιο σημαντικό στοιχείο, καθώς περιέχει το υπόλοιπο του εγγράφου και γίνεται συνώνυμο με τον τύπο εγγράφου. Δεν μπορεί να επαναληφθεί.

Τα έγγραφα markup, όπως τα HTML, XML ή SGML, εισάγουν ένα στοιχείο αφετηρίας το οποίο περιλαμβάνει όλα τα υπόλοιπα. Το στοιχείο αφετηρίας συνήθως περιγράφει τη εστία ή τη λειτουργία του εγγράφου. Το HTML στοιχείο στην HTML είναι ένα καλό στοιχείο αφετηρίας επειδή αποκαλύπτει το όνομα της γλώσσας markup. Το στοιχείο TRIVIA στο παράδειγμα είναι ένα άλλο στοιχείο αφετηρίας επειδή περιγράφει τη λειτουργία του εγγράφου.

Όλα τα XML στοιχεία περιλαμβανομένων και των στοιχείων αφετηρίας έχουν μια συγκεκριμένη σύνταξη. Περιλαμβάνουν το όνομά τους ή τον τύπο τους ανάμεσα στα σύμβολα < και >, τα οποία χρησιμοποιούνται συχνά σε HTML, SGML και XML έγγραφα. Για παράδειγμα, το στοιχείο της παραγράφου στην XML γράφεται ως <P>. Υπάρχουν άλλοι κανόνες για τα στοιχεία και άλλοι για τα χαρακτηριστικά, τα οποία περιέχονται στα στοιχεία, αλλά τα περισσότερα στοιχεία παίρνουν τη μορφή: <NAME>CONTENT</NAME>. Η XML είναι case sensitive, οπότε ένα στοιχείο μπορεί να εμφανίζεται ως: <DocumentTitle>XML Tutorial</DocumentTitle>.

Το δικό σας παράδειγμα της well-formed XML

Είσατε τώρα στο στάδιο της δημιουργίας ενός δικού σας εγγράφου στο οποίο πρέπει να αποφασίσετε για το περιεχόμενό του και τα στοιχεία που θα χρησιμοποιήσετε στην περιγραφή του. Διαλέξτε ένα στοιχείο που να περιγράφει το έγγραφο που θα δημιουργήσετε. Χρησιμοποιήστε αυτό το στοιχείο ως στοιχείο αφετηρίας, καθώς περιέχει όλα τα στοιχεία μέσα σε αυτό, και περιγράψτε το έγγραφό σας. Για παράδειγμα, τα NOVEL, SONG και LOG είναι καλά στοιχεία αφετηρίας επειδή υποδηλώνουν τη λειτουργία ολόκληρου του εγγράφου. Αυτό το έγγραφο περιέχει βοήθεια για τη δημιουργία των XML εγγράφων και χρησιμοποιεί το <HELP> ως κατάλληλα επιλεγμένο στοιχείο αφετηρίας.

```
<?xml version="1.0" standalone="yes"?>
<HELP>

<HELP/>
```

Παράδειγμα εγγράφου

Αυτό το έγγραφο είναι ένα καλό παράδειγμα μιας ιδανικής περιγραφής περιεχομένου χρησιμοποιώντας XML. Ξεκινάει με μια XML δήλωση και εισάγει ένα καλό στοιχείο αφετηρίας. Περιγράφει, επίσης, καλά τη δομή ολόκληρου του εγγράφου, επιβεβαιώνοντας ότι παρουσιάζει καλά τις διαφορές μεταξύ διαφόρων τύπων του περιεχομένου.

```
<?xml version="1.0" standalone="yes"?>
<TRIVIAL>

<MATH>
<QUESTION>What is the square root of 25</QUESTION>
<ANSWER>5</ANSWER>
</MATH>

<GENERAL>
<QUESTION>What is the season after Summer</QUESTION>
<ANSWER>Autumn</ANSWER>
</GENERAL>

</TRIVIA>
```

ΒΗΜΑ 3: Γράφοντας σε XML

Σκοπός αυτού του βήματος είναι να περιγράψουμε το παράδειγμα της well-formed XML που δημιουργήσατε. Θα πρέπει να γράψετε το έγγραφό σας και να το περιγράψετε με XML markup, όπως απορρέει. Θα είναι το μεγαλύτερο βήμα από τη στιγμή που δημιουργήσατε το σώμα του εγγράφου.

Τώρα που έχετε μια εστία για το έγγραφό σας αναπτύξτε το. Ξεκινήστε να γράφετε περιγράφοντας το περιεχόμενο με στοιχεία της δικής σας ονομασίας. Επινοήστε καινούριο στοιχείο κάθε φορά που εισάγετε περιεχόμενο στο έγγραφό σας και που να διαφέρει σημαντικά από κάθε προηγούμενο. Όσο πιο βαθιά είναι η περιγραφή του εγγράφου, τόσο μεγαλύτερο έλεγχο θα έχετε αργότερα, όταν το χρησιμοποιείτε, προφανώς όταν γράφετε ένα stylesheet.

Μπορείτε επίσης να χρησιμοποιείτε χαρακτηριστικά για να περιγράψετε το έγγραφό σας. Περιέχονται μέσα στα start-tags των στοιχείων και χρησιμοποιούνται για να περιγράψουν το περιεχόμενό τους ή την συμπεριφορά τους. Για παράδειγμα, ένα χαρακτηριστικό χρησιμοποιείται για να περιγράψει τη διαφορά μεταξύ των ταξινομημένων και των μη-ταξινομημένων λιστών, εάν όλες οι λίστες περιγραφόταν με το στοιχείο <LIST>. Αν το χαρακτηριστικό ονομαζόταν type, το start-tag μιας μη-ταξινομημένης λίστας θα έπαιρνε τη μορφή: <LIST type = "unordered">. Μπορείτε να χρησιμοποιήσετε τα στοιχεία και τα χαρακτηριστικά ως μέρος της σειράς εργαλείων.

Χρησιμοποίησε παραδείγματα XML, όπως την υπόδειξη TRIVIA στο βήμα 2, το οποίο περιγράφει ερωτήσεις και απαντήσεις και τη συνομιλία σε αυτό το βήμα. Πρόσεξε ότι και τα δυο παραδείγματα αναπαριστούν συστατικά εγγράφου με περιγραφικά στοιχεία. Αποτελεσματικές XML υποδείξεις χρησιμοποιούν περιγραφικά ονόματα στοιχείων με τέτοιο τρόπο που επιβεβαιώνει ότι τα έγγραφα είναι εύκολο να δημιουργηθούν και να διατηρηθούν.

Οι συγγραφείς ονομάζουν τυπικά τα έγγραφά τους. Τα στοιχεία TITLE και TTL χρησιμοποιούνται συχνά σε έγγραφα ονομασίας και σε συστατικά εγγράφων, όπως πίνακες και φιγούρες. Δεν είναι υποχρεωτικό να περιλαμβάνεις πάντα έναν τίτλο εγγράφου ωστόσο εάν χρησιμοποιηθεί παρέχει μια πολύ καλή περιγραφή στους αναγνώστες. Το όνομα του εγγράφου πρέπει να βρίσκεται στην αρχή του εγγράφου μετά το στοιχείο αφετηρίας.

Το δικό σας παράδειγμα της well-formed XML

Σε αυτό το βήμα πρέπει να συμπληρώσετε το δικό σας παράδειγμα. Το επόμενο παράδειγμα θα σας βοηθήσει να κατανοήσετε κάθε βήμα αυτού του tutorial. <?xml

```
<version="1.0" standalone="yes"?>
<HELP>
<TITLE><XML Help</TITLE>

<QUERY area="XML">
<QUESTION>I don't know where to start. What do II do?</QUESTION>
```

```

<ANSWER>Start with our root element. Break your document down into parts, filling them in and flushing them
out further</ANSWER>
</QUERY>

<QUERY area="XML">
<QUESTION>How do I know if my element names are well chosen?</QUESTION>
<ANSWER>If their purpose is implied through their names, then they are well chosen. Make sure, at the very
least, that they make sense to you. If not rename them or change your structure.</ANSWER>
</QUERY>

<QUERY area="formatting">
<QUESTION>Where can I learn about formetting XML documents?</QUESTION>
<ANSWER>You should learn much more about XML before you concern yourself with formatting. Formating is an
external process and is best done with XSL, CSS or DSSSL.</ANSWER>
</QUERY>

</HELP>

```

Το παράδειγμα ενός XML εγγράφου που περιγράφει μια συζήτηση.

Αυτό το έγγραφο περιγράφει μια συζήτηση μεταξύ δυο ανθρώπων. Μπορείτε να το χρησιμοποιήσετε ως ακόμη ένα παράδειγμα μιας καλής περιγραφής κειμένου. Η χρήση των κεφαλαίων για τα ονόματα στοιχείων και των μικρών γραμμάτων για τα ονόματα των χαρακτηριστικών είναι μια πρόταση που δεν είναι υποχρεωτικό να ακολουθηθεί.

```

<version="1.0" standalone="yes"?>
<CONVERSATION>
<TITLE>A Conversation between Richard and Annie</TITLE>

<SPEECH>
<PERSON name="Annie">Why XML?</PERSON>
< PERSON name="Richard">XML conforms to information allowing document authors to create markup languages
that work for them.</ PERSON>
</SPEECH>

<SPEECH>
<PERSON name="Richard">With HTML, document authors become frustrated, trying to fit their information
sets into a fixed markup language.</ PERSON>
< PERSON name="Annie">For authors that have become used to HTML, but only want a few more elements, can
they use XML?< /PERSON>
< PERSON name="Richard">With XML, anything is possible. You could extend or even contract HTML, depending
on needs. You can also define your own markup language, to describe any organization of information. Think of XML
as being information-dependent.</ PERSON>
</SPEECH>

</CONVERSATION>

```

Βήμα 4: Έλεγχος και ανάλυση του εγγράφου.

Στόχος αυτού του βήματος είναι να γίνει ο έλεγχος για το αν ακολουθεί το έγγραφο τους κανόνες της well-formed XML. Εάν το έγγραφό σας δεν συμφωνεί με αυτούς τους κανόνες δεν μπορεί να διαβαστεί κανονικά από έναν XML reader.

Εάν το έγγραφό σας δεν είναι πολύ καλά μορφοποιημένο μπορεί να μην ενεργεί όπως είναι επιθυμητό. Επιβεβαιώστε ότι το κάθε στοιχείο έχουν από μια ετικέτα αρχής και τέλους (start and end tags) και ειδικά το στοιχείο της αφετηρίας. Τα start και end tags όλων των στοιχείων πρέπει να φωλιάζουν ιδανικά μέσα σε υψηλότερα στοιχεία, που σημαίνει ότι τα στοιχεία δεν μπορούν να επικαλυφθούν. Επίσης, έλεγξε ότι όλα τα χαρακτηριστικά χρησιμοποιούν μονά ή διπλά εισαγωγικά και ότι η τιμή κάθε χαρακτηριστικού περιέχεται από έναν μόνο τύπο χαρακτηριστικών.

Ένας καλύτερος τρόπος ελέγχου εάν το έγγραφο είναι well-formed είναι η χρήση του parser (αναλυτή), ο οποίος ελέγχει για λάθη το XML έγγραφο. Προσφέρει μεγάλη βοήθεια όταν δεν μπορείτε να βρείτε το λόγο για τον οποίο τα

έγγραφα σας δεν διαβάζονται κανονικά. Ένας τρόπος να ελέγξεις την καλή μορφοποίηση του εγγράφου είναι να χρησιμοποιήσεις την εντολή NSGMLS: NSGMLS -s xml.dcl yourfile.xml. Για αυτόν το σκοπό υπάρχουν πολλοί άλλοι καλοί parsers.

Παραδείγματα ελέγχου και ανάλυσης εγγράφων

Αυτά τα παραδείγματα επεξηγούν κοινά μυστικά και την ιδανική χρήση των στοιχείων. Τα πρώτα δυο παραδείγματα είναι σε well-formed XML ενώ τα υπόλοιπα που ακολουθούν δεν είναι παρόλο που ελέγχουν για λάθη.

```
<PRICE>$57.80</PRICE>
<PET><CAT type="Corish Rex">Cat nestes properly within PET.</CAT><PET>
<WEATHER>foggy
<LEVEL>intermediate<LEVEL>
<PASSWORD>planetB612</PASSWD>
<DISTANCE TYPE=KM120</DISTANCE>
<CAR><engine>engine does not net properly within CAR</CAR></engine>
```

2.1.2 Valid XML.

Αυτό το Tutorial είναι σχεδιασμένο σαν εισαγωγή στη δημιουργία valid XML έγγραφων. Το Tutorial για well formed XML είναι καλή προϋπόθεση για αυτό το tutorial, αφού όλα τα έγγραφα XML, well formed και valid, πρέπει να είναι τουλάχιστον well formed. Χρησιμοποιώντας αυτό το tutorial, θα μάθετε πώς να γράφετε valid XML, και να δημιουργήσετε έγγραφα valid XML. Αν μπερδευτείτε, κοιτάξτε τα παραδείγματα που βρίσκονται σε αυτό το tutorial ή κάντε μια ανασκόπηση στο tutorial για well formed XML.

Η βασική διαφορά μεταξύ της valid και well formed XML είναι η σχέση τους με το DTD(Document Type Definition, Ορισμός Τύπου Εγγράφου). Η well formed XML είναι σχεδιασμένη για χρήση χωρίς DTD, ενώ η valid XML ρητά απαιτεί χρήση DTD. Ένα DTD είναι ένα σετ κανόνων που ένα έγγραφο ακολουθεί, και τους οποίους το software μπορεί να πρέπει να διαβάσει προτού επεξεργαστεί και εμφανίσει το έγγραφο. Αυτοί οι κανόνες γενικά καθορίζουν το όνομα και το περιεχόμενο κάθε στοιχείο (element) από τα οποία το απόσπασμα μπορεί και πρέπει να αποτελείται. Το στοιχείο παράγραφος μπορεί να οριστεί ότι περιέχει στοιχείο λέξη κλειδί (elements keyword), και κώδικα και ότι υπάρχει μέσα σε ενότητα στοιχείων (elements section) και σημείωση (note). Τα valid XML έγγραφα έχει συγκεκριμένα προοδευτικά χαρακτηριστικά γνωρίσματα, στα οποία δεν μπορούν να έχουν πρόσβαση τα well formed έγγραφα, λόγω της έλλειψης τους σε DTD. Αυτά τα χαρακτηριστικά γνωρίσματα μπορούν να βελτιώσουν σημαντικά την χρησιμότητα του εγγράφου συμπεριλαμβανομένου του: μηχανισμού link, οντότητες (entities) και χαρακτηριστικά (attributes). Τα περισσότερα XML Web sites είναι πιθανώς φτιαγμένα από valid XML έγγραφα, συμμορφωμένα (conforming) με συνήθη DTD, επιτρέποντας στους δημιουργούς τους να δομούν ελεύθερα τα sites τους και να χρησιμοποιούν δημιουργικότερα σετ χαρακτηριστικών από ότι η HTML παραδοσιακά επέτρεπε.

Τα valid XML έγγραφα προσφέρουν πολύ περισσότερα στην επεξεργασία εγγράφου από ότι τα αντίστοιχα well formed. Η συγγραφή, επεξεργασία, αποθήκευση, και εμφάνιση του εγγράφου έγιναν ευκολότερες γιατί το έγγραφο υπάρχει σε ένα δομικό περιβάλλον. Οι συγγραφείς πρέπει να δημιουργήσουν έγγραφα με βάση μια προκαθορισμένη δομή επωφελούμενοι από την καθαρό μοντέλο εγγράφου. Όπως και τα well-formed XML, έτσι και τα valid έγγραφα πρέπει να συνοδεύονται από style sheets για να πετύχουν αισθητική εμφάνιση. Η χρήση των CSS, XSL, ή DSSSL style sheets για valid έγγραφα είναι όπως και στα well-formed έγγραφα, εκτός από το ότι τα style sheets μπορεί να φτιαχτούν σύμφωνα με ένα DTD, καθώς εναντιώνεται σε μια σχετικά άγνωστη χρήση του markup.

Παρακάτω ακολουθούν τα βήματα στα οποία μπορείτε να βασιστείτε για τη σύνταξη valid XML εγγράφων.

Βήμα 1: Κατανόηση του DTD (Document Type Definition, Ορισμός Τύπου Εγγράφου).

Το DTD είναι το θεμέλιο των Valid XML εγγράφων, παρέχοντας τον ορισμό του τύπου του εγγράφου, για τα έγγραφα που θα ακολουθήσουν. Το DTD περιέχει τις απαραίτητες πληροφορίες για την εγγραφή και επεξεργασία του εγγράφου. Χωρίς αυτές τις πληροφορίες, οι αναγνώστες του εγγράφου μπορεί να μην ξέρουν πώς να επεξεργαστούν links, εικόνες ή οντότητες και οι συγγραφείς του εγγράφου θα στερούνται ένα περίγραμμα για την ανάπτυξη.

Προτού σχεδιάσετε ένα DTD θα πρέπει να έχετε σαφή γνώση του τι τύπου έγγραφο θα δημιουργήσετε. Θα πρέπει να διαλέξετε τύπο εγγράφου και να τον ονομάσετε με ονόματα όπως: novel, memo, webpage, letter ή report. Αφού διαλέξετε τον τύπο του εγγράφου σας, μπορείτε να αρχίσετε να «χτίζετε» το δικό σας DTD, επεκτείνοντας το έτσι ώστε να ταιριάζει με τις απαιτήσεις του εγγράφου σας.

Το DTD εμπεριέχει ένα μικρό συντακτικό που μπορεί να μαθευτεί γρήγορα. Αυτό το συντακτικό έχει πολλά στοιχεία αλλά μπορούν να αθροιστούν σε δυο απαραίτητες δομές, οι οποίες είναι το στοιχείο (element) και το χαρακτηριστικό (attribute). Αυτές οι δυο δομές στο έγγραφο χρησιμοποιούνται για να περιγράψουν το περιεχόμενο. Η χρήση τους σε ένα έγγραφο πρέπει να ορίζεται σε ένα DTD για να εξασφαλίζεται ότι τα conforming έγγραφα είναι valid. Το στοιχείο αφετηρίας (element root) είναι το πιο σημαντικό και περιέχει όλα τα άλλα στοιχεία. Αρχίζετε λοιπόν με αυτό, ορίζετε το περιεχόμενο του, μετά ορίζετε τα στοιχεία που είναι στο περιεχόμενο του έως ότου φτάσετε στα στοιχεία επιπέδου κειμένου. Η διαδικασία ορισμού των χαρακτηριστικών (attribute) δεν είναι τόσο κυκλική, απλώς απαιτεί τον ορισμό των χαρακτηριστικών για κάθε στοιχείο (element) που χρησιμοποιεί χαρακτηριστικά. Τα στοιχεία (elements) παίρνουν γενικά την μορφή: <ELEMENT NAME CONTENT>. Τα χαρακτηριστικά (attributes) συχνά παίρνουν την μορφή: <!ATTLIST ELEMENT-NAME NAME CDATA #IMPLIED>.

Παράδειγμα DTD για έγγραφο τύπου Novel (μυθιστόρημα).

Αυτό το DTD περιγράφει τη δομή ενός απλού novel (μυθιστόρημα). Ο τύπος εγγράφου αυτού του DTD είναι novel, όπως μπορείτε να δείτε από το στοιχείο αφετηρίας (root element) που ορίζεται στην πρώτη γραμμή. Το DTD ορίζει για κάθε στοιχείο, το όνομα του και το περιεχόμενο του, και ορίζει και κάθε στοιχείο του περιεχομένου του έως ότου το DTD να είναι πλήρως καθορισμένο. Τα στοιχεία είναι το μόνο επιτρεπτό περιεχόμενο για πολλά από τα στοιχεία αυτού του DTD, και δεν μπορούν να περιέχουν κείμενο απ'ευθείας. Το περιεχόμενο «#PCDATA» σε κάποια από τα στοιχεία σημαίνει ότι επιτρέπεται κείμενο ή χαρακτήρας δεδομένο.

```
<ELEMENT novel (preface,chapter+,biography?,criticalessay*)>
<ELEMENT preface (paragraph+)>
<ELEMENT chapter (title,paragraph+,section+)>
<ELEMENT section (title,paragraph+)>
<ELEMENT biography (title,paragraph+)>
<ELEMENT criticalessay (title,section+)>
<ELEMENT paragraph (#PCDATA| Keyword)*>
<ELEMENT title (#PCDATA| keyword)*>
<ELEMENT keyword (#PCDATA)*>
```

Βήμα 2: Γράφοντας ένα DTD.

Ο στόχος αυτού του βήματος είναι η δημιουργία DTD. Η συγγραφή ενός DTD μπορεί να είναι δύσκολη, αλλά ακολουθώντας τις οδηγίες που περιγράφονται εδώ, βλέποντας τα παραδείγματα και κρατώντας το DTD σας απλό, θα μπορείτε να γράψετε το δικό σας DTD.

Τα DTD είναι φτιαγμένα από δηλώσεις markup, που είναι τύπου στοιχείο (element), λίστα χαρακτηριστικών (attribute list), οντότητας (entity) και notation δηλώσεις. Αυτές οι δηλώσεις καθορίζουν την δομή, παραστατικά, και αποθηκεύουν τις πληροφορίες για ένα τύπο εγγράφου. Σου επιτρέπουν να περιγράψεις ποικίλα κομμάτια του τύπου του εγγράφου σου, ούτως ώστε οι συγγραφείς να μπορούν να παράγουν conforming (προσαρμοσμένα) έγγραφα με ευκολία. Όπως είπαμε στο Βήμα 1 οι δηλώσεις τύπου element και attribute list είναι τα πιο σημαντικές και πιο συχνά χρησιμοποιούμενες δηλώσεις. Αυτό το tutorial θα επικεντρωθεί σε αυτές τις δηλώσεις, μια και είναι πιο χρήσιμες και σχετικά πιο εύκολες για μάθηση.

Οι δηλώσεις για τύπο element ορίζουν τον τύπο των στοιχείων, που έχουν ήδη οριστεί στον markup, σε έγγραφο, σαν element. Κάθε τύπος στοιχείου (element) περιέχει ένα όνομα (name), περιεχόμενο (content) και πιθανώς ένα σετ από χαρακτηριστικά (attribute) και μπορούν να αποφασισθούν πολλές φορές σαν προσαρμοσμένα (conforming) στοιχεία αυτού του τύπου. Το περιεχόμενο του στοιχείου μπορεί να έχει τέσσερις μορφές, από τις οποίες η πιο κοινή είναι mixed-και element-περιεχόμενο. Τα μοντέλα element-περιεχόμενο φτιάχνουν λίστα από γκρουπ στοιχείων και γκρουπ από γκρουπ στοιχείων με συγκεκριμένες σχέσεις μεταξύ τους, για να καθορίσουν το συγκεκριμένο περιεχόμενο των τύπων element. Τα σειριακά γκρουπ περιγράφουν μια απαιτούμενη και τακτική εμφάνιση των μελών τους που χωρίζονται από το σύμβολο «,». Τα γκρουπ επιλογών περιγράφουν μια μόνη εμφάνιση ενός μόνο από τα μέλη τους που χωρίζονται με το σύμβολο «|». Η εμφάνιση κάθε ενός από τα μέλη αυτού του γκρουπ και του ίδιου του γκρουπ μπορεί να αλλάξει από τις παρακάτω ενδείξεις εμφάνισης, οι οποίες ορίζουν την εμφάνιση σαν: «+», απαραίτητη και επαναληπτή, «*», προαιρετική και επαναληπτή και «?» προαιρετική. Δηλώσεις τύπου element παίρνουν την παρακάτω μορφή: <ELEMENT NAME CONTENT>. Το παρακάτω παράδειγμα περιέχει ένα τύπο element, όνομα και σαν περιεχόμενο, ένα σειριακό γκρουπ και φωλιασμένο προαιρετικό-επαναληπτέο γκρουπ επιλογών: <ELEMENT EXAMPLE (TITLE?, PARA, (PARA | NOTE |

CODE) *». Αυτή η δήλωση τύπου element περιγράφει ένα τύπο element που μπορεί να περιέχει ένα TITLE, πρέπει να περιέχει ένα PARA και που μπορεί να περιέχει κάποιο αριθμό από μη τακτική εμφάνιση από PARA, NOTE και CODE.

Οι δηλώσεις για τύπου element Mixed-περιεχόμενο είναι αρκετά παρόμοιες με τις αντίστοιχες για element-περιεχόμενο αλλά διαφέρει σε κάποια χαρακτηριστικά. Πρέπει να χρησιμοποιούν ένα προαιρετικό-επαναληπτικό γκρουπ επιλογών και επίσης πρέπει να περιέχει #PCDATA, κείμενο (text), σαν πρώτο μέλος τους. Τύποι Mixed-περιεχομένου element που δεν περιέχει element-περιεχόμενο αλλά μόνο κείμενο δεν χρησιμοποιεί ένα προαιρετικό-επαναληπτικό γκρουπ επιλογών. Οι δυο παρακάτω δηλώσεις για τύπο element mixed-περιεχομένου που επεξηγούν τις συντακτικές διαφορές μεταξύ element τύπου mixed-περιεχομένου που περιέχει element-περιεχόμενο και αυτών που δεν περιέχουν:

<!ELEMENT PARAGRAPH (#PCDATA | Keyword) *> <!ELEMENT Keyword (#PCDATA)>.

Οι δηλώσεις attribute είναι πιο ευμετάβλητες από τις δηλώσεις τύπου element, έχοντας δέκα διαφορετικούς τύπους. Οι πιο κοινοί και απλοί τύποι attribute είναι τα string και enumeration (κατάλογος) attributes. Τα attribute τύπου string, κοινώς ονομαζόμενα CDATA attribute, σας επιτρέπουν να «συλλάβετε» εικονικά αυθόρμητα κείμενα string για να περιγράψουν στοιχείο (element) περιεχόμενο ή συμπεριφορά. Τα enumeration attributes είναι παρόμοια με τα χαρακτηριστικά (attribute) τύπου string αλλά απαιτούν να θέσετε μια λίστα από επιλογές για να μπορεί να κάνει ο συγγραφέας. Και τα δυο αυτά χαρακτηριστικά (attributes) απαιτούν attributes defaults, που μπορεί να είναι: #REQUIRED, απαιτούμενο, #IMPLIED, προαιρετικό, #FIXED "value", μια καθορισμένη τιμή και , "value", μια default αλλά παραβιασθεα τιμή. Τα χαρακτηριστικά (attribute) τύπου string παίρνουν τη μορφή: <!ATTLIST ELEMENT-NAME NAME CDATA DEFAULT>. Τα enumeration attributes παίρνουν την μορφή: <!ATTLIST ELEMENT-NAME NAME (choice1 | choice2 | choicen) DEFAULT>. Και οι δυο τύποι χαρακτηριστικών μετατρέπονται στην markup στην παρακάτω μορφή, στην ετικέτα αρχής στοιχείου (element start-tags): , ELEMENT-NAME NAME="value">.

Η δική σας Valid XML υπόδειξη

Θα πρέπει να «χτίσετε» το δικό σας XML έγγραφο, καθώς διαβάσετε αυτό το tutorial. Κοιτάξτε τη δική σας Valid XML υπόδειξη καθώς διαβάσετε το tutorial, για καθοδήγηση και σαν παράδειγμα. Το δικό σας έγγραφο μπορεί να είναι κάπως διαφορετικό από αυτό το παράδειγμα αλλά θα πρέπει να έχει «χτιστεί» με τον ίδιο τρόπο. Σε αυτό το σημείο της δημιουργίας εγγράφου XML, δεν έχετε καθόλου έγγραφο, αλλά το DTD με το οποίο θα συνδέεται το έγγραφο. Προσέξτε πως αυτό το παράδειγμα χρησιμοποιεί δηλώσεις τύπου element για να συλλάβει το περιεχόμενο του εγγράφου και δηλώσεις τύπου attributes για να περιγράψει συγκεκριμένα τμήματα του.

```
<!ELEMENT MEMO (TO, FROM, SUBJECT,BODY, SIGN) >
<!ATTLIST MEMO importance (HIGH | MEDIUM | LOW) "LOW" >
<ELEMENT TO (#PCDATA)>
<ELEMENT FROM (#PCDATA)>
<ELEMENT SUBJECT (#PCDATA)>
<ELEMENT BODY (P+)>
<ELEMENT P (#PCDATA)>
<ELEMENT SIGN (#PCDATA)>
<!ATTLIST SIGN signatureFile CDATA #IMPLIED
email CDATA #REQUIRED>
```

Χειριστές ακολουθίας και επιλογής, και ενδείξεις εμφάνισης.

Οι παρακάτω χειριστές ακολουθίας και επιλογής, και ενδείξεις εμφάνισης τροποποιεί την φύση του τύπου element.

Όρος	Ερμηνεία
,	Χειριστής ακολουθίας που χωρίζει μέλη τις λίστες ακολουθίας, που απαιτεί ακολουθιακή χρήση όλων των μελών.
	Χειριστής επιλογής που χωρίζει μέλη της λίστας επιλογής, που απαιτεί την χρήση ενός και μόνο από τα μέλη. Αυτό το non-symbol δείχνει εμφάνιση.
+	Αυτό το σύμβολο δείχνει μια απαραίτητη και επαναληπτέα εμφάνιση
*	Αυτό το σύμβολο δείχνει προαιρετική και επαναληπτέα εμφάνιση.
?	Αυτό το σύμβολο δείχνει προαιρετική εμφάνιση.

Βήμα 3: Κάντε μια XML δήλωση.

Ο στόχος αυτού του βήματος είναι να δημιουργήσετε μια XML δήλωση, που δηλώνει την φύση του εγγράφου XML στον αναγνώστη του εγγράφου. Χωρίς αυτή τη δήλωση, ένας αναγνώστης εγγράφου μπορεί να αναγνωρίσει λάθος ένα XML έγγραφο σαν SGML, HTML ή RTF, προκαλώντας προβλήματα ερμηνείας.

Κάθε XML έγγραφο πρέπει να χρησιμοποιεί μια XML δήλωση, για να εξηγήσει την φύση του σε αναγνώστες XML εγγράφων. Editors, browsers και επεξεργαστές εγγράφων χρησιμοποιούν την δήλωση για να καθορίσουν πως θα πρέπει ένα έγγραφο να επεξεργαστεί. Η δήλωση γίνεται όλο και περισσότερο απαραίτητη με μεγάλα και πολύπλοκα έγγραφα αλλά πρέπει επίσης να χρησιμοποιείται και σε μικρότερα και δοκιμαστικά έγγραφα. Η δήλωση XML περιλαμβάνει πληροφορίες πάνω στην markup language (γλώσσα), την έκδοση της markup γλώσσας, την παρουσία εξωτερικών markup δηλώσεων και κωδικοποίηση χαρακτήρων.

XML Δήλωση.

Αυτές οι XML δηλώσεις χρησιμοποιούνται γενικά για ποικίλους τύπους XML συγγραφής. Οι πρώτες δυο δηλώσεις μπορούν να χρησιμοποιηθούν για να περιγράψουν well-formed και valid XML έγγραφα, αντίστοιχα. Η τρίτη δήλωση μπορεί να θεωρηθεί default XML δήλωση, δηλώνοντας ότι είναι έγγραφο XML έκδοσης 1.0, και ότι δεν μπορεί να σταθεί από μόνο του (standalone) χωρίς εξωτερικές markup δηλώσεις και ότι είναι κωδικοποιημένο σε UTF-8, μια 8-bit Unicode κωδικοποίησης χαρακτήρων.

```
<?xml version="1.0" standalone="yes"?>
<?xml version="1.0" standalone="no"?>
<?xml version="1.0" standalone="no" encoding="UTF-8"?>
```

Η δική σας Valid XML υπόδειξη

Μπορεί να έχετε μπερδευτεί και να αναρωτιέστε που πήγε το DTD. Θεωρείστε ότι είναι σωσμένο σε άλλο αρχείο και αναμένει να χρησιμοποιηθεί. Είναι πολύ σημαντικό το να χτίσετε το DTD πριν γράψετε το έγγραφο αλλά το έγγραφο πρέπει να υποστεί προκαταρκτική ανάπτυξη πριν μπορέσει να χρησιμοποιήσει το DTD. Το έγγραφο σας πρέπει να αρχίζει με μια XML δήλωση και πρέπει να είναι παρόμοιο με αυτό το έγγραφο σε αυτό προ πρωταρχικό στάδιο της ανάπτυξης του.

```
<?xml version="1.0" standalone="no"?>
```

Συστατικά XML δήλωσης.

Όρος	Ερμηνεία
<?xml	Αρχίζει την δήλωση XML, ένα είδος οδηγίας για την επεξεργασία
version	Η δήλωση version περιγράφει την έκδοση της XML που χρησιμοποιείται, που πρέπει να είναι ίση με "1.0", καθώς η XML 1.0 είναι η τωρινή και μόνη έκδοση της XML.
standalone	Η Standalone δήλωση εγγράφου επιτρέπει στον συγγραφέα του εγγράφου να διευκρινίζει αν υπάρχουν εξωτερικές markup δηλώσεις. Αυτό το χαρακτηριστικό μπορεί να είναι ίσο με yes ή no και είναι συνήθως το τελευταίο στα valid έγγραφα.
encoding	Η encoding δήλωση επιτρέπει στον συγγραφέα να καθορίσει τον χαρακτήρα κωδικοποίησης που θα χρησιμοποιηθεί. Αυτή η δήλωση πρέπει να χρησιμοποιείται μόνο από συγγραφείς που χρησιμοποιούν χαρακτήρα κωδικοποίησης εκτός του US-ASCII, τον πιο κοινό, ή τον UTF-8.
?>	Κλείνει την XML δήλωση.

Βήμα 4: Περιλαμβάνοντας μια δήλωση τύπου εγγράφου (DTD)

Σκοπός αυτού του βήματος είναι να περιλάβετε μια δήλωση τύπου εγγράφου στο έγγραφο σας, το οποίο θα οδηγεί (link) ή θα περιλαμβάνει το δικό σας DTD. Η δήλωση τύπου εγγράφου (DTD) ονομάζει τον τύπο του εγγράφου που χρησιμοποιείται και οδηγεί (link) ή περιλαμβάνει τον ορισμό του, το DTD.

Η δήλωση τύπου εγγράφου, που τοποθετείται μετά την δήλωση XML, είναι ένας μηχανισμός για την ονομασία του τύπου του εγγράφου με τον οποίο συμμορφώνεται το έγγραφο και δείχνει ή περιλαμβάνει τον ορισμό του. Τα Valid XML έγγραφα πρέπει να δηλώνουν τον τύπο εγγράφου με τον οποίο συμμορφώνονται και παρέχουν τον πλήρη ορισμό του έτσι ώστε οι editors, οι browsers και οι μετατροπείς (converters) να μπορούν να διαβάσουν το DTD του για να το κατανοήσουν. Τα well-formed έγγραφα μπορούν επίσης να περιλάβουν δήλωση τύπου εγγράφου και να περιλάβουν markup δηλώσεις στα εξωτερικά τους subset αλλά δεν είναι απαραίτητο να πράξουν έτσι. Η δήλωση τύπου εγγράφου ονομάζει του τύπο εγγράφου κάνοντας αναφορά στο στοιχείο αφετηρίας (root element) του εγγράφου. Μπορεί να κάνει

αναφορά σε ένα εξωτερικό DTD, που καλείται **external DTD subset**, να περιλαμβάνει το DTD εσωτερικά στο **internal DTD subset**, ή και τα δυο. Η δήλωση τύπου εγγράφου παίρνει γενικά την μορφή: `<!DOCTYPE NAME SYSTEM "file"[]>`.

Η χρήση της δήλωσης τύπου εγγράφου είναι πολύτιμη, εξαρτάται από το DTD υποσύνολο (subset) που χρησιμοποιείται για να περιέχει το DTD. Το DTD μπορεί να «στεγαστεί» αποκλειστικά είτε σε εξωτερικό (external), είτε σε εσωτερικό (internal) υποσύνολο, ή και στα δυο. Σε κάθε μια από αυτές τις επιλογές μπορεί να χρησιμοποιηθεί ελαφρώς διαφορετική σύνταξη. Η πιο απλή δήλωση, η οποία επιτρέπει μόνο εξωτερικό (external) υποσύνολο (subset) του DTD που ακολουθεί: `<!DOCTYPE NAME SYSTEM "file">`. Η ακόλουθη δήλωση επιτρέπει την χρήση ενός από τους δύο τύπους υποσυνόλων ή και τους δυο και είναι και ποιο γνωστή: `<!DOCTYPE NAME SYSTEM "FILE" []>`. Το [] στεγάζει το εσωτερικό υποσύνολο. Η τελευταία δήλωση επιτρέπει μόνο την χρήση εσωτερικού υποσυνόλου: `<!DOCTYPE NAME []>`.

Αν το στοιχείο αφετηρίας (root element) είναι ένα συγκεκριμένο DTD το ροem τότε η δήλωση τύπου εγγράφου θα διαβάσει `<!DOCTYPE roem []>`. Ο ορισμός του ροem τύπου εγγράφου θα πρέπει να περιλαμβάνεται σε ένα εσωτερικό DTD υποσύνολο, μαρκαρισμένο από τους [και το] δείκτες. Αυτός ο ορισμός θα μπορούσε επίσης να σωθεί σε ένα εξωτερικό αρχείο και να συνδεθεί (link) με την δήλωση του τύπου εγγράφου στην παρακάτω μορφή: `<!DOCTYPE roem SYSTEM "roem.dtd">`.

Η δική σας Valid XML υπόδειξη

Θα πρέπει να προσθέσετε μια δήλωση τύπου εγγράφου στο δικό σας XML έγγραφο, που θα ονομάζει το τύπο του εγγράφου σας, θα δείχνει ίσως στο εξωτερικό υποσύνολο ή θα περιλαμβάνει ένα εσωτερικό υποσύνολο, το οποίο θα είναι μαρκαρισμένο από τους δείκτες [και]. Αυτό το παράδειγμα ονόμασε τον τύπο του εγγράφου σαν MEMO, δεν δείχνει σε εξωτερικό υποσύνολο, αλλά περιλαμβάνει markup δηλώσεις σε ένα εσωτερικό υποσύνολο. Το DTD θα μπορούσε το ίδιο απλά να αποθηκευτεί σαν ένα εξωτερικό αρχείο και να χρησιμοποιήσει την παρακάτω δήλωση τύπου εγγράφου: `<!DOCTYPE MEMO SYSTEM "memo.dtd">`.

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE MEMO [
<ELEMENT MEMO          (TO,FROM,SUBJECT,BODY,SIGN)>
<ATTLIST MENO           importance    (HIGH | MEDIUM | LOW) "LOW">
<ELEMENT TO             (#PCDATA)>
<ELEMENT FROM           (#PCDATA)>
<ELEMENT SUBJECT        (#PCDATA)>
<ELEMENT BODY           (P+)>
<ELEMENT P              (#PCDATA)>
<ELEMENT SIGN           (#PCDATA)>
<ATTLIST SIGN           signatureFile CDATA #IMPLIED
                        email          CDATA #REQUIRED>
]>
```

Όρος	Ερμηνεία
<!DOCTYPE	Αρχίζει την δήλωση τύπου του εγγράφου.
NAME	Ονομάζει τον τύπο του εγγράφου όπως σαφώς ορίστηκε, και πρέπει να ακολουθεί τους κανόνες XML NAME.
SYSTEM	Δείχνει ότι ένας system identifier, που ακολουθεί πρέπει να διαβαστεί και να αναλυθεί.
"report.dtd"	Αναφορά σε system identifier, που πρέπει να αναλυθεί για να εντοπισθεί το DTD.
[Αρχίζει ένα εσωτερικό υποσύνολο. Το εσωτερικό DTD υποσύνολο, που μπορεί να περιέχει αριθμό markup δηλώσεων.
]	Τελειώνει το εσωτερικό υποσύνολο.
>	Κλείνει τη δήλωση τύπου εγγράφου.

Βήμα 5: Γράφοντας Valid XML.

Ο σκοπός αυτού το βήματος είναι να γράψετε ένα valid XML έγγραφο, σιγουρεύοντας ότι είναι συμμορφωμένο με το DTD σας. Αφού κοπιάσαμε με το DTD, το γράψιμο του εγγράφου είναι εύκολο. Η Valid XML είναι πραγματικά ευκολότερη στο γράψιμο από ότι η well formed XML γιατί απλά ακολουθείς τη δομή του DTD, ο οποίο απαιτεί πολύ λιγότερες επιλογές.

Η συγγραφή Valid XML εγγράφου είναι πολύ διαφορετική από ότι του well-formed XML. Πρέπει να ακολουθείτε το DTD που εσείς γράψατε, σιγουρεύοντας ότι δεν θα σπάσετε κανένα από τους κανόνες που εσείς γράψατε. Αν το έγγραφο σας γίνει μη-συμμορφωμένο (non-conformant), καλά θα κάνετε να το φτιάξετε, γιατί αλλιώς πιθανώς να μην εμφανιστεί ή επεξεργαστεί σωστά όταν το χρησιμοποιήσετε. Το έγγραφο σας πρέπει να περιέχει δηλώσεις XML και τύπου εγγράφου. Βαλτέ ετικέτες αρχής και τέλους για το στοιχείο αφετηρίας (root element) σας. Ακολουθήστε το DTD επακριβώς, δίνοντας προσοχή στους κανόνες περιεχομένου για κάθε στοιχείο, σιγουρεύοντας ότι συμμορφώνεστε εκεί που απαιτείται και να επιλέγετε εκεί που επιτρέπεται.

Τα λάθη είναι ένα μεγάλο πρόβλημα για αυτούς που αναπτύσσουν έγγραφα. Αν προσέξετε το γράψιμο σας και σιγουρευτείτε ότι ακολουθείτε το DTD, θα αποφύγετε τα λάθη. Βεβαιωθείτε ότι τα στοιχεία σας είναι φωλιασμένα σωστά, καθώς overlapping στοιχεία προκαλούν λάθη. Αν βρείτε ότι το DTD σας είναι πολύ αυστηρό για το έγγραφο που θέλετε να γράψετε τότε τροποποιήστε το DTD σας. Το DTD είναι δικό σας και μπορεί να τροποποιηθεί αν χρειαστεί. Κρατείστε την συνολική δομή του εγγράφου στο μυαλό σας, σιγουρεύοντας ότι η απόδοση του σε XML είναι σωστή και βγάζει νόημα.

Η δική σας Valid XML υπόδειξη

```
<? xml version="1.0" standalone="no"?>

<!DOCTYPE MEMO [
<!ELEMENT MEMO          (TO,FROM,SUBJECT,BODY,SIGN)>
<!ATTLIST  MEMO          importance    (HIGH | MEDIUM | LOW) "LOW">
<!ELEMENT TO            (#PCDATA)>
<!ELEMENT FROM          (#PCDATA)>
<!ELEMENT SUBJECT       (#PCDATA)>
<!ELEMENT BODY          (P+)>
<!ELEMENT P             (#PCDATA)>
<!ELEMENT SIGN          (#PCDATA)>
<!ATTLIST  SIGN          signatureFile CDATA #IMPLIED
                    email             CDATA #REQUIRED>
]>

<MEMO importance="HIGH">
<TO>Tutorial Tarkers</TO>
<FROM>Tutorial Writer</FROM>
<SUBJECT>Your impressions</SUBJECT>
<BODY>
<P>Now that you are almost done the tutorial, you must be getting an idea of what XML is about. These
emerging technologies sometimes take time thou before they catch on.</P>
<P>Did you find the tutorial helpful? Which areas did you find confusing? How would you improve them?</P>
</BODY>
<SIGN email="rlander@pdbeam.uwaterloo.ca">Richard</SIGN>
</MEMO>
```

Βήμα 6: Αναλύοντας του δικό σας Valid XML έγγραφο.

Ο σκοπός του βήματος αυτού είναι να βεβαιωθείτε ότι το έγγραφο σας είναι well-formed και valid. Αν το έγγραφο σας δεν είναι συμμορφωμένο με τους κανόνες της XML και του DTD σας, μπορεί να μην διαβάζεται σωστά από έναν XML αναγνώστη. Τα λάθη μπορεί να καταστρέψουν την δουλειά που έχετε κάνει στο έγγραφο και μπορεί να απαιτούν πολύ δουλειά για να θεραπεύσουν το πρόβλημα ειδικά αν είναι error-dense. Τα περισσότερα valid XML έγγραφα είναι φτιαγμένα σε validating XML editors, οι οποίοι είναι σχεδιασμένοι για να αποφεύγουν λάθη και συχνά να μην τα επιτρέπουν.

Αφού επενδύσετε το χρόνο σας για να δημιουργήσετε ένα XML έγγραφο, θα περιμένετε να συμπεριφέρεται σωστά. Αν το έγγραφο σας δεν είναι πλήρως well-formed, μπορεί να μην φέρεται όπως θα θέλαμε. Βεβαιωθείτε ότι όλα τα στοιχεία (element) έχουν ετικέτες αρχής και τέλους, ειδικά το στοιχείο αφετηρίας (root element) σας. Όλες οι ετικέτες τέλους και αρχής των στοιχείων πρέπει να φωλιάζουν σωστά σε υψηλότερα στοιχεία, που σημαίνει ότι τα στοιχεία δεν μπορούν να παρακάμπτουν, όπως με τα car και engine elements στο παράδειγμα παρακάτω. Επίσης τσεκάρετε αν όλα τα

attributes σας χρησιμοποιούν μόνο ή δίπλα εισαγωγικά και ότι κάθε τιμή χαρακτηριστικά (attribute) περιέχεται μέσα σε ένα μόνο τύπο εισαγωγικών.

Ένας καλύτερος, και πιο ακριβής τρόπος έλεγχου του πόσο well-formed και valid είναι το valid XML έγγραφο σας είναι η χρήση ενός αναλυτή (parser), ο οποίος ελέγχει το έγγραφο σας για λάθη. Σιγουρευτείτε ότι χρησιμοποιείτε validating XML parser αν ελέγχετε valid XML έγγραφα. Οι parsers μπορεί να γίνουν πολύ εξυπηρετικοί όταν δεν μπορείτε να βρείτε το λόγο γιατί το έγγραφο δεν διαβάζεται σωστά. Είναι επίσης πολύ καλό εργαλείο για τον έλεγχο των εγγράφων σας όπως τα δημιουργείτε και πριν τα χρησιμοποιήσετε. Ο James Clark παρέχει ένα εύχρηστο parser, που καλείτε NSGMLS, που περιλαμβάνεται στο SP και Jake packages του. μπορείτε να χρησιμοποιήσετε την ακόλουθη NSGMLS εντολή για να ελέγξετε το κατά πόσο το έγγραφο σας είναι well-formed και valid: NSGMLS -s xml.dcl yourfile.xml. Πολλοί parser υπάρχουν για αυτό το σκοπό.

Κοινά λάθη ανάλυσης (parsing).

Αυτά τα παραδείγματα parsing δείχνουν κοινά λάθη και σωστή χρήση elements. Τα πρώτα δυο παραδείγματα είναι well-formed XML, ενώ τα ακόλουθα είναι από αυτά που θα παράγουν parsing λάθη, καθώς δεν είναι well-formed. Μοντελοποιήστε τα elements σας σύμφωνα με τα πρώτα δυο παραδείγματα, ακολουθώντας τους κανόνες well-formed. Βεβαιωθείτε ότι το markup σας συμφωνεί με το DTD σας, καθώς το κατά πόσο το έγγραφο σας είναι valid είναι σχεδόν το ίδιο σημαντικό με τον αν είναι well-formed στα valid XML έγγραφα.

```
<PRICE>$57.80</PRICE>
<PET><CAT type="Corish Rex">Cat nestes properly within PET.</CAT><PET>
<WEATHER>foggy
<LEVEL>intermediate<LEVEL>
<PASSWORD>planetB612</PASSWD>
<DISTANCE TYPE=KM120</DISTANCE>
<CAR><engine>engine does not net properly within CAR</CAR></engine>
```

2.2 Χαρακτήρες (Characters)

Μια αναλυμένη οντότητα περιέχει **κείμενο**, ως ακολουθία χαρακτήρων, η οποία μπορεί να αντιπροσωπεύει δεδομένα χαρακτήρων ή δεδομένα markup (markup or character data). Ο χαρακτήρας είναι μια ατομική μονάδα κειμένου όπως καθορίστηκε από το ISO/IEC 10646. Νόμιμοι χαρακτήρες είναι το κενό(tab), ο χαρακτήρας επιστροφής (carriage return), η γραμμή ανατροφοδότησης(line feed) καθώς και οι νόμιμοι γραφικοί χαρακτήρες του UNICODE και του ISO/IEC 10646. Η χρήση των συμβατών χαρακτήρων έχει αποτραπεί.

Character Range

[2] Char ::= #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF] /* any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. */

Ο μηχανισμός κωδικοποίησης χαρακτήρων σε bit πρότυπα ποικίλει από οντότητα σε οντότητα. Όλοι οι XML επεξεργαστές θα πρέπει να αποδεχθούν τις κωδικοποιήσεις UTF-8 και UTF-16 του 10646.

2.3 Κοινά Συντακτικά Εργαλεία (Common Syntactic Constructs)

Παρακάτω παρατίθενται σύμβολα που χρησιμοποιούνται ευρύτατα στην γραμματική.

s (white space) περιέχει έναν ή περισσότερους κενούς(#x20) χαρακτήρες, τον χαρακτήρα επιστροφής, την γραμμή ανατροφοδότησης και τα tabs.

White Space

[3] S ::= (#x20 | #x9 | #xD | #xA)+

Οι χαρακτήρες ταξινομούνται για περισσότερη εμπιστοσύνη ως γράμματα, ψηφία ή άλλοι χαρακτήρες. Τα γράμματα αποτελούνται από μια αλφαβητική ή συλλαβική βάση χαρακτήρων που πιθανόν ακολουθούνται από έναν ή περισσότερους συνδυασμούς χαρακτήρων ή από έναν ιδεογραφικό χαρακτήρα.

Το **Name** είναι μια ένδειξη που ξεκινάει με ένα γράμμα ή με έναν χαρακτήρα στίξης, και συνεχίζει με γράμματα, ψηφία, παύλες, υπογραμμίσεις, σύμβολα ή τελείες γνωστά ως χαρακτήρες ονόματος (name characters). Τα names που ξεκινούν με το string "xml" ή οποιοδήποτε string που να ταιριάζει (('X' 'x') ('M' 'm') ('L' 'l')), είναι δεσμευμένα για τυποποίηση σε αυτήν την έκδοση ή σε μελλοντικές αυτής της προδιαγραφής.

ΣΗΜΕΙΩΣΗ: ο χαρακτήρας «σύμβολο»(colon character) στα names της XML είναι δεσμευμένο για πειραματισμούς με τα με name spaces. Η έννοια αυτή αναμένεται να τυποποιηθεί στο μέλλον σε σημείο που τα έγγραφα που χρησιμοποιούν το σύμβολο για πειραματικούς σκοπούς να πρέπει να εκσυγχρονιστούν. (Δεν υπάρχει καμία εγγύηση ότι οποιοσδήποτε μηχανισμός name space που υιοθετήθηκε για την XML θα χρησιμοποιεί στην πραγματικότητα το σύμβολο ως χαρακτήρα αρχής ή τέλους). Στην πράξη, αυτό σημαίνει ότι οι συγγραφείς δεν θα χρησιμοποιούν το colon στα names της XML παρά μόνον ως μέρος των πειραμάτων name space, αλλά αυτοί οι επεξεργαστές XML θα πρέπει να αποδεχτούν το colon ως name character.

Το Nmtoken (name token) αποτελεί μια μίξη των name character.

Names and Tokens

[4]	NameChar	::=	Letter Digit '.' '-' '_' ':' CombiningChar Extender
[5]	Name	::=	(Letter '_' ':') (NameChar)*
[6]	Names	::=	Name (S Name)*
[7]	Nmtoken	::=	(NameChar)+
[8]	Nmtokens	::=	Nmtoken (S Nmtoken)*

Το literal data είναι οποιοδήποτε string ανάθεσης τιμής που δεν περιλαμβάνει τη χρήση των εισαγωγικών ως χαρακτήρα αρχής ή τέλους για αυτό το string. Τα literals χρησιμοποιούνται για να καθορίσουν τις τιμές των εσωτερικών οντοτήτων/internal entities(EntityValue), τις τιμές των χαρακτηριστικών/value of attributes(AttValue) και τους εξωτερικούς αναγνωριστές/external identifiers (SystemLiteral). Προσέξτε ότι ένα SystemLiteral μπορεί να αναλυθεί χωρίς έλεγχο για markup.

Literals

[9]	EntityValue	::=	'" ([^%&"] PEReference Reference)* "' '" ([^%&'] PEReference Reference)* '"
[10]	AttValue	::=	'" ([^<&"] Reference)* "' '" ([^<&'] Reference)* '"
[11]	SystemLiteral	::=	('"' [^"]* '"') ('"' [^']* '"')
[12]	PubidLiteral	::=	'" PubidChar* "' '" (PubidChar - '"')* '"
[13]	PubidChar	::=	#x20 #xD #xA [a-zA-Z0-9] [-'()+,./:=?;!*#@\$_%]

2.4 Χαρακτήρες Δεδομένων & Markup (Character Data and Markup)

Το κείμενο(text) αποτελείται από ένα συνδυασμό character data & markup. Το **markup** παίρνει τη μορφή των χαρακτήρων αρχής(start tags), τέλους(end tags), του κενού(empty element tags), των οντοτήτων(entity references), των χαρακτήρων(character references), των σχολίων(comments), των χαρακτήρων αρχής ή τέλους σε ένα CDATA section(CDATA section delimiters), των δηλώσεων του κειμένου(document type declarations) και των οδηγιών επεξεργασίας(processing instructions).

Το κείμενο που δεν είναι markup αποτελεί τον **character data** του εγγράφου.

Οι χαρακτήρες &, < μπορούν να εμφανιστούν στην αληθινή τους μορφή μόνον όταν χρησιμοποιούνται σαν χαρακτήρες markup αρχής ή τέλους, ή μέσα σε σχόλιο, σε οδηγίες επεξεργασίας, ή σε CDATA section. Για να εμφανισθούν αλλού πρέπει να χρησιμοποιηθούν είτε σε αναφορές αριθμητικών χαρακτήρων ή αντιστοίχως σε string "&" και "<". Το σύμβολο > μπορεί να αναπαρασταθεί χρησιμοποιώντας το string ">".

Στα περιεχόμενα των στοιχείων, τα δεδομένα χαρακτήρων είναι οποιοδήποτε string στοιχείων που δεν περιλαμβάνουν τον χαρακτήρα αρχής από οποιοδήποτε markup. Σε ένα CDATA section τα δεδομένα χαρακτήρων είναι οποιοδήποτε string χαρακτήρων που δεν περιλαμβάνουν χαρακτήρα κλεισίματος αρχής ή τέλους της CDATA section, "]]>".

Για να επιτραπεί στις τιμές των χαρακτηριστικών να περιέχουν και μονά και διπλά εισαγωγικά, η απόστροφος ή μονά εισαγωγικά (') μπορούν να αναπαρασταθούν ως "'" ενώ τα διπλά εισαγωγικά (") ως """.

Character Data

[14] CharData ::= [^&]* - ([^&]* ']'>' [^&]*)

2.5 Σχόλια (Comments)

Τα **σχόλια** μπορούν να εμφανιστούν οπουδήποτε στο κείμενο έξω από το markup; επιπροσθέτως, μπορεί να εμφανιστούν στην περιοχή δηλώσεων του εγγράφου όπου επιτρέπεται από την γραμματική. Δεν είναι μέρος των χαρακτήρων δεδομένων; ένας XML επεξεργαστής δίνει τη δυνατότητα σε μια εφαρμογή να ανακτήσει το κείμενο των σχολίων. Για συμβατότητα, ο χαρακτήρας "- "(διπλή παύλα) δεν θα πρέπει να υπάρχει μεταξύ των σχολίων.

Ένα παράδειγμα σχολίου είναι το εξής:

```
<!-- declarations for <head> & <body> -->
```

2.6 Οδηγίες Επεξεργασίας (Processing Instructions)

Οι **οδηγίες επεξεργασίας** (PI) επιτρέπουν στα έγγραφα να περιέχουν οδηγίες για τις διάφορες εφαρμογές

Processing Instructions

[16] PI ::= '<?' PITarget (S (Char* - (Char* '>' Char*)))? '>'

[17] PITarget ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))

Οι οδηγίες επεξεργασίας δεν είναι μέρος των character data των εγγράφων, αλλά πρέπει να διέρχονται μέσα από τις εφαρμογές. Ξεκινούν με έναν στόχο -target (PITarget) που χρησιμοποιείται να προσδιορίσει την εφαρμογή στην οποία απευθύνεται η οδηγία. Τα target names "XML", "xml" είναι δεσμευμένα για τυποποίηση σε αυτήν ή σε μελλοντικές εκδόσεις αυτής της προδιαγραφής. Ο **XML μηχανισμός σχολίων** μπορεί να χρησιμοποιείται στις τυπικές δηλώσεις των PI targets

2.7 Τμήματα CDATA (CDATA Sections)

Τα **CDATA Sections** μπορούν να λάβουν χώρα οπουδήποτε υπάρχουν character data. Χρησιμοποιούνται να αποφεύγουν κείμενο το οποίο περιέχει χαρακτήρες οι οποίοι αναγνωρίζονται ως markup. Τα CDATA Sections ξεκινούν με τον συμβολισμό "<![CDATA[" και τελειώνουν με τον εξής "]]>".

CDATA Sections

[18] CDSECT ::= CDStart CData CDEnd

[19] CDStart ::= '<![CDATA['

[20] CData ::= (Char* - (Char* ']]>' Char*))

[21] CDEnd ::= ']]>'

Σε ένα CDATA Section, μόνον το string CDEnd αναγνωρίζεται ως markup, έτσι ώστε τα αριστερά άγκιστρα και το σύμβολο & να υπάρχουν στην πραγματική τους μορφή δεν μπορούν να αποφευχθούν χρησιμοποιώντας "<" και "&". Τα CDATA Sections δεν μπορούν να φωλιαστούν.

Ένα παράδειγμα από CDATA Section στο οποίο τα "<greeting>" και "</greeting>" αναγνωρίζονται ως character data και όχι ως markup:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

2.8 Δήλωση Εγγράφων (Prolog and Document Type Declaration)

Τα XML έγγραφα ξεκινούν με μια **XML δήλωση** η οποία καθορίζει την έκδοση της XML που χρησιμοποιείται. Για παράδειγμα το παρακάτω είναι ένα XML έγγραφο καλά αναπτυγμένο αλλά όχι έγκυρο:

```
<?xml version="1.0"?>
<greeting>Hello, world!</greeting>
```

Όπως και αυτό:

```
<greeting>Hello, world!</greeting>
```

Η έκδοση νούμερο "1.0" πρέπει να χρησιμοποιείται για να υποδηλώσει συμμόρφωση στην έκδοση αυτής της προδιαγραφής; είναι λάθος για ένα έγγραφο να χρησιμοποιεί την έκδοση 1 εάν δεν συμβαδίζει με την έκδοση αυτής της προδιαγραφής. Ήταν στόχος της ομάδας εργασίας της XML να δώσει σε επόμενες εκδόσεις αυτής της προδιαγραφής αριθμούς διαφορετικούς της μονάδας, αλλά αυτή η πρόθεση δεν υποδηλώνει υπόσχεση να παραχθούν νέες μελλοντικές εκδόσεις της XML, ούτε ότι αν παραχθούν να χρησιμοποιηθεί συγκεκριμένος τρόπος αρίθμησης. Εφόσον δεν αποκλείονται μελλοντικές εκδόσεις, αυτό το σχέδιο παρέχεται ως μέσο για να εμφανιστεί η πιθανότητα της αυτόματης

αναγνώρισης έκδοσης, εάν αυτό κριθεί απαραίτητο. Οι επεξεργαστές μπορεί να εμφανίσουν μήνυμα λάθους εάν λαμβάνουν έγγραφα άλλης έκδοσης την οποία δεν υποστηρίζουν.

Η λειτουργία του markup σε ένα XML έγγραφο είναι να περιγράφει την αποθηκευμένη μνήμη και τη λογική δομή και να συνδέει ζεύγη απόδοσης τιμής με τις λογικές δομές του. Η XML παρέχει έναν μηχανισμό, τη δήλωση εγγράφου (document type declaration) για να καθορίζει τους περιορισμούς σε μια λογική δομή και να υποστηρίξει τη χρήση των προκαθορισμένων μονάδων αποθήκευσης. Ένα έγγραφο XML είναι έγκυρο(valid) εάν έχει μια συνδεδεμένη δήλωση εγγράφου και αν το έγγραφο συμφωνεί με τους περιορισμούς που εκφράζονται σε αυτό.

Το document type declaration πρέπει να εμφανίζεται πριν από το πρώτο στοιχείο του εγγράφου.

Prolog

```
[22] prolog ::= XMLDecl? Misc* (doctypeddecl Misc*)?
[23] XMLDecl ::= '<?xml' VersionInfo EncodingDecl? SDDDecl? S? '>'
[24] VersionInfo ::= S 'version' Eq (' VersionNum ' | " VersionNum ")
[25] Eq ::= S? '=' S?
[26] VersionNum ::= ([a-zA-Z0-9_.:] | '-' )+
[27] Misc ::= Comment | PI | S
```

Το XML **document type declaration** περιέχει δηλώσεις markup που παρέχουν γραμματική για μια κατηγορία εγγράφων. Αυτή η γραμματική είναι γνωστή ως document type definition, ή DTD. Η δήλωση εγγράφων δίνει έμφαση σε ένα εξωτερικό υποσύνολο (μια ειδική κατηγορία εξωτερικής οντότητας) περιέχοντας δηλώσεις markup, ή μπορεί να περιέχει δηλώσεις markup απ' ευθείας σε ένα εσωτερικό υποσύνολο, ή και τα δυο. Το DTD για ένα έγγραφο αποτελείται και από τα δυο υποσύνολα μαζί.

Μια δήλωση markup είναι δήλωση τύπου στοιχείου (element type declaration), δήλωση λίστας χαρακτηριστικών (attribute-list declaration), δήλωση οντότητας (entity declaration) ή δήλωση σχολίων (notation declaration). Αυτές οι δηλώσεις μπορεί να περιέχονται ως σύνολο ή ως μέρος μέσα στις οντότητες παραμέτρων (parameter entities), όπως περιγράφονται παρακάτω.

Document Type Definition

```
[28] DocTypedDecl ::= '<!DOCTYPE' S Name (S ExternalID)? S? ('['
[                VC: Root Element Type ]
                (markupdecl | PReference | S)* ']' S)? '>'
[29] Markupdecl ::= elementdecl | AttlistDecl | EntityDecl | NotationDecl |
[                VC: Proper
                PI | Comment
                Declaration/PE Nesting ]
[                WFC: PEs in Internal
                Subset ]
```

Οι δηλώσεις markup μπορεί να επινοούνται ως σύνολο ή ως μέρος του κειμένου αντικατάστασης (replacement text) ή των οντοτήτων παραμέτρων (parameter entities).

Περιορισμός Valid: Μοντέλο Τύπου Βάσης

Το Name στη δήλωση τύπου εγγράφου πρέπει να ταιριάζει με τον τύπο στοιχείου του στοιχείου βάσης.

Περιορισμός Valid: Ιδανική Δήλωση / PE Φύλλισμα

Οι παράμετροι οντότητας του κειμένου αντικατάστασης πρέπει να είναι κατάλληλα φωλιασμένες με δηλώσεις markup. Με άλλα λόγια, εάν είτε ο πρώτος χαρακτήρας είτε ο τελευταίος χαρακτήρας μιας δήλωσης markup (markupdecl παραπάνω) περιέχεται σε ένα κείμενο αντικατάστασης για μια αναφορά παραμέτρων οντοτήτων (parameter entity reference), και οι δυο θα πρέπει να περιέχονται στο ίδιο κείμενο αντικατάστασης.

Περιορισμός Well-formed: PE σε εσωτερικά υποσύνολα

Στο εσωτερικό DTD υποσύνολο, οι αναφορές παραμέτρων οντοτήτων μπορούν να πραγματοποιηθούν μόνον όπου υπάρχουν δηλώσεις markup και όχι μέσα σε αυτές. (Αυτό δεν εφαρμόζεται σε αναφορές που πραγματοποιούνται σε εξωτερικές παραμέτρους οντοτήτων ή σε εξωτερικά υποσύνολα).

Όπως και στα εσωτερικά υποσύνολα, τα εξωτερικά υποσύνολα και οποιαδήποτε εξωτερική παράμετρος οντοτήτων που αναφέρονται στο DTD πρέπει να περιλαμβάνει ένα σύνολο ολοκληρωμένων δηλώσεων markup των τύπων που επιτρέπονται από τον συμβολισμό markupdecl, και να διασπείρονται με το κενό ή τις αναφορές παραμέτρων οντοτήτων. Παρ' όλα αυτά τμήματα των περιεχομένων των εξωτερικών παραμέτρων οντοτήτων ή των εξωτερικών υποσυνόλων μπορούν να αγνοηθούν χρησιμοποιώντας την υποθετική κατασκευή τμήματος; αυτό δεν επιτρέπεται στα εσωτερικά υποσύνολα..

External Subset

[30] extSubset ::= TextDecl? extSubsetDecl

[31] extSubsetDecl ::= (markupdecl | conditionalSect | PEReference | S)*

Τα εξωτερικά υποσύνολα και οι εξωτερικές παράμετροι οντοτήτων διαφέρουν επίσης από τα εσωτερικά υποσύνολα στο γεγονός ότι σε αυτά οι αναφορές παραμέτρων οντοτήτων επιτρέπονται *εντός* των δηλώσεων markup και όχι μόνον *ανάμεσά* τους.

Ένα παράδειγμα ενός XML εγγράφου με ένα document type declaration:

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world</greeting>
```

Ο αναγνωριστής του συστήματος "hello.dtd" δίνει το URI του εγγράφου DTD.

Οι δηλώσεις μπορούν επίσης να δίνονται τοπικά όπως στο παράδειγμα:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
<!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world</greeting>
```

Εάν χρησιμοποιούνται και εξωτερικά και εσωτερικά υποσύνολα, το εσωτερικό υποσύνολο πραγματοποιείται πριν το εξωτερικό υποσύνολο. Αυτό έχει ως αποτέλεσμα ότι οι οντότητες και οι δηλώσεις λίστας χαρακτηριστικών στο εσωτερικό υποσύνολο προηγούνται αυτών στο εξωτερικό υποσύνολο.

2.9 Δήλωση Εγγράφων (Standalone Document Declaration)

Οι δηλώσεις markup μπορούν να επηρεάσουν το περιεχόμενο του εγγράφου, καθώς αυτό περνά από έναν XML επεξεργαστή σε μια εφαρμογή; τέτοια παραδείγματα είναι η απουσία χαρακτηριστικών και οι δηλώσεις οντοτήτων. Αυτού του είδους η δήλωση, που μπορεί να εμφανιστεί ως χαρακτηριστικό των XML δηλώσεων, επισημαίνει ή όχι την ύπαρξη δηλώσεων που εμφανίζονται εξωτερικά της οντότητας του εγγράφου.

Standalone Document Declaration

[32] SDDDecl ::= S 'standalone' Eq (("'" ('yes' | 'no') "'") | (''" ('yes' | 'no') "'") [VC:Standalone Document Declaration]

Η τιμή "yes" σε αυτές τις δηλώσεις δείχνει ότι δεν υπάρχουν δηλώσεις markup εξωτερικά της οντότητας του εγγράφου που να επηρεάζουν τις πληροφορίες που διέρχονται από τον XML επεξεργαστή στην εφαρμογή. Η τιμή "no" δείχνει ότι μπορεί να υπάρχουν τέτοιες εξωτερικές δηλώσεις markup. Αυτού του είδους η δήλωση εξασφαλίζει την παρουσία των εξωτερικών δηλώσεων; η παρουσία, σε ένα έγγραφο, αναφορών σε εξωτερικές οντότητες, όταν αυτές οι οντότητες δηλώνονται στο εσωτερικό, δεν αλλάζουν αυτήν την standalone ιδιότητα..

Εάν δεν υπάρχουν εξωτερικές δηλώσεις markup, η μορφή δηλώσεων standalone δεν έχει καμία σημασία. Εάν υπάρχουν εξωτερικές δηλώσεις markup, αλλά δεν υπάρχει η μορφή δηλώσεων standalone, υποθέτουμε ότι έχουμε την τιμή "no".

Οποιοδήποτε XML έγγραφο για το οποίο ισχύει standalone = "no" μπορεί να μετατραπεί αλγοριθμικά σε ένα standalone έγγραφο το οποίο μπορεί να είναι επιθυμητό σε κάποιες δικτυακές εφαρμογές.

Περιορισμός Valid: Δήλωση Εγγράφων Standalone

Αυτού του είδους η δήλωση εγγράφων πρέπει να παίρνει την τιμή "no" εάν οποιαδήποτε εξωτερική δήλωση markup περιέχει τις εξής δηλώσεις:

- Χαρακτηριστικά με default τιμές, εάν τα στοιχεία στα οποία αποδίδονται αυτά τα χαρακτηριστικά εμφανίζονται στο έγγραφο χωρίς λεπτομερή παρουσίαση των τιμών για αυτά τα χαρακτηριστικά, ή
- Οντότητες (εκτός των amp, lt, gt, apos, quot), εάν εμφανίζονται στο έγγραφο αναφορές για αυτές τις οντότητες, ή
- Χαρακτηριστικά με τιμές που υποβάλλονται σε κανονικοποίηση, όπου το χαρακτηριστικό εμφανίζεται στο έγγραφο με τιμή η οποία αλλάζει ως αποτέλεσμα της κανονικοποίησης, ή
- Στοιχειώδης τύπους με στοιχειώδη περιεχόμενα, εάν το κενό (white space) υπάρχει ανάμεσα σε οποιαδήποτε ύπαρξη αυτών των τύπων.

Ένα παράδειγμα είναι το εξής:

```
<?xml version="1.0" standalone="yes"?>
```

2.10 Χειρισμός White Space (Handling White Space)

Συντάσσοντας XML έγγραφα είναι συχνά αξιόπιστο να χρησιμοποιούνται «λευκά κενά» ("white space") (όπως κενά, tabs, λευκές γραμμές) για να υπερέχει το markup έχοντας περισσότερη αναγνωσιμότητα.. Αυτά τα λευκά κενά δεν προτείνονται για συνυπολογισμό της ελεύθερης έκδοσης του εγγράφου. Από την άλλη μεριά, τα «σημαντικά» λευκά κενά που πρέπει να εμφανίζονται στην ελεύθερη έκδοση είναι κοινά, για παράδειγμα στην ποίηση και στον πηγαίο κώδικα.

Ένας XML επεξεργαστής πρέπει να περάσει όλους τους χαρακτήρες του εγγράφου που δεν είναι markup στην εφαρμογή. Μάλιστα ένας νόμιμος XML επεξεργαστής πρέπει να ενημερώσει την εφαρμογή για το ποιοι από αυτούς τους χαρακτήρες αποτελούνται από λευκά κενά τα οποία εμφανίζονται στα στοιχειώδη περιεχόμενα.

Ένα ειδικό χαρακτηριστικό που ονομάζεται xml: space μπορεί να επισυναφθεί σε ένα στοιχείο για να επισημάνει ότι σε αυτό το στοιχείο το λευκό κενό θα διατηρείται από τις εφαρμογές. Σε valid έγγραφα, αυτό το χαρακτηριστικό όπως και οποιοδήποτε άλλο θα πρέπει να δηλώνεται πριν χρησιμοποιηθεί. Όταν δηλώνεται θα πρέπει να δίνεται ως ένας απριθμητός τύπος του οποίου οι πιθανές τιμές είναι οι "default" και "preserve". Για παράδειγμα:

```
<!ATTLIST poem xml:space (default|preserve) 'preserve'>
```

Η τιμή "default" δηλώνει ότι η μέθοδος επεξεργασίας του λευκού κενού της εφαρμογής είναι αποδεκτή από αυτό το στοιχείο; η τιμή "preserve" δείχνει ότι οι εφαρμογές διατηρούν όλο το λευκό κενό. Αυτή η δηλωμένη πρόθεση πρέπει να εφαρμόζεται σε όλα τα στοιχεία μέσα στο περιεχόμενο του στοιχείου στο οποίο υπάρχει, εκτός αν υπερέχει με ένα άλλο παράδειγμα του XML χαρακτηριστικού.

Το στοιχείο της κορυφής οποιουδήποτε εγγράφου δεν έχει κανέναν σκοπό εκτός εάν παρέχει τιμή σε αυτό το χαρακτηριστικό ή εάν το χαρακτηριστικό είναι δηλωμένο με τιμή default.

2.11 Χειρισμός End-Of-Line (End Of Line Handling)

Οι αναλυμένες XML οντότητες αποθηκεύονται συχνά σε αρχεία υπολογιστών και για περισσότερη αξιοπιστία στην σύνταξη, οργανώνονται σε γραμμές. Αυτές οι γραμμές χωρίζονται τυπικά από συνδυασμούς των χαρακτήρων carriage return(#xD) και line-feed(#xA).

Για να απλοποιηθεί η αποστολή των εφαρμογών, σε μια εξωτερικά αναλυμένη οντότητα ή στην αληθινή αξία μιας εσωτερικά αναλυμένης οντότητας ή στην αληθινή αλληλουχία δυο χαρακτήρων "#xD#xA" ή σε ένα αληθινό standalone "#xD", ο XML επεξεργαστής θα πρέπει να περάσει στην εφαρμογή τον μονό χαρακτήρα #xA.(αυτή η συμπεριφορά μπορεί αξιόπιστα να παραχθεί κανονικοποιώντας όλα τα line breaks σε #xA κατά την εισαγωγή πριν την ανάλυση.

2.12 Αναγνώριση της Γλώσσας (Language Identification)

Στην επεξεργασία του εγγράφου, είναι συχνά χρήσιμο να προσδιορίζεται η φυσική ή τυπική γλώσσα στην οποία είναι γραμμένο το περιεχόμενο. Ένα ειδικό χαρακτηριστικό που ονομάζεται xml : lang μπορεί να εισαχθεί στα έγγραφα για να καθορίσει τη γλώσσα που χρησιμοποιείται στα περιεχόμενα και στις αξίες των χαρακτηριστικών οποιουδήποτε στοιχείου σε ένα XML έγγραφο. Στα valid έγγραφα αυτό το χαρακτηριστικό θα πρέπει να δηλώνεται πριν χρησιμοποιηθεί. Οι τιμές του χαρακτηριστικού είναι οι αναγνωριστές γλώσσας όπως καθορίζονται από το [IETF RFC 1766], "Tags For The Identification Of Languages".

Language Identification

[33]	LanguageID	::=	Langcode ('-' Subcode)*
[34]	Langcode	::=	ISO639Code IanaCode UserCode
[35]	ISO639Code	::=	([a-z] [A-Z]) ([a-z] [A-Z])
[36]	IanaCode	::=	('i' 'I') '-' ([a-z] [A-Z])+
[37]	UserCode	::=	('x' 'X') '-' ([a-z] [A-Z])+
[38]	Subcode	::=	([a-z] [A-Z])+

Το langcode μπορεί να είναι οποιοδήποτε από τα επόμενα:

- Ένας κώδικας γλώσσας δυο γραμμάτων όπως καθορίζεται από το [ISO 639], "Codes For The Representation Of Names Of Languages"
- Έναν αναγνωριστή γλώσσας όπως καταγράφηκε από το Internet Assigned Numbers Authority [IANA]; αυτό ξεκινάει με το πρόθεμα "-i"(ή "-I").

- Έναν αναγνωριστή γλώσσας που καθορίζεται από τον χρήστη ή είναι σύμφωνος κάποιων συμμετοχών για ιδιωτική χρήση; αυτά ξεκινούν με το πρόθεμα "-x" ή "-X" με την πρόθεση ότι δεν συγκρούονται με τυποποιημένα ή δεσμευμένα names από τον IANA.

Μπορεί να υπάρχει οποιοσδήποτε αριθμός τμημάτων subcode; αν υπάρχει το πρώτο τμήμα υποκώδικα και ο υποκώδικας αποτελείται από δυο γράμματα, τότε πρέπει να υπάρχει ένας κώδικας από τον [ISO 3166], "Codes For The Representation Of Names Of Countries". Αν ο πρώτος υποκώδικας αποτελείται από περισσότερα από ένα γράμματα, πρέπει να υπάρχει ένας υποκώδικας για τη γλώσσα σε μορφή ερώτησης όπως καθορίζεται από τον IANA, εκτός εάν το langcode ξεκινάει με το πρόθεμα πρόθεμα "-x" ή "-X".

Είναι σύνηθες να δίνουν στον κώδικα γλώσσας χαμηλότερη συνθήκη και στον κώδικα country (εάν υπάρχει) υψηλότερη. Πρόσεξε ότι αυτές οι τιμές σε αντίθεση με άλλα names είναι case insensitive.

Για παράδειγμα:

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
<sp who="Faust" desc='leise' xml:lang="de">
<l>Habe nun, ach! Philosophie,</l>
<l>Juristerei, und Medizin</l>
<l>und leider auch Theologie</l>
<l>durchaus studiert mit heißem Bemüh'n.</l>
</sp>
```

Ο σκοπός που δηλώνεται ως xml:lang εφαρμόζεται σε όλα τα χαρακτηριστικά και στο περιεχόμενο των στοιχείων όπου έχει καθοριστεί, εκτός αν υπερισχύσει ένα άλλο στοιχείο μέσα στο περιεχόμενο.

Μια απλή δήλωση για xml:lang μπορεί να πάρει τη μορφή:

```
Xml:lang NMTOKEN #IMPLIED
```

Αλλά αν αρμόζει μπορούν να δίδονται και συγκεκριμένες default τιμές. Για μια συλλογή γαλλικών ποιημάτων για Άγγλους μαθητές, με σχόλια και σημειώσεις στα αγγλικά, το χαρακτηριστικό xml:lang μπορεί να δηλωθεί ως εξής:

```
<!ATTLIST poem xml:lang NMTOKEN 'fr'>
<!ATTLIST gloss xml:lang NMTOKEN 'en'>
<!ATTLIST note xml:lang NMTOKEN 'en'>
```

3. Λογικές Δομές (Logical Structures)

Κάθε XML έγγραφο περιέχει ένα ή περισσότερα στοιχεία (elements), τα όρια των οποίων καθορίζονται από start-tags και end-tags, για κενά(empty) στοιχεία, από έναν empty element tag. Κάθε στοιχείο έχει έναν τύπο που καθορίζεται από το όνομά του, που μερικές φορές ονομάζεται generic identifier(GI), και που μπορεί να έχει ένα σύνολο χαρακτηριστικών προδιαγραφής. Κάθε χαρακτηριστικό προδιαγραφής έχει ένα όνομα(name) και μια τιμή(value).

Element

[39]	element	::=	EmptyElemTag	
			STag content ETag	[WFC: Element Type Match]
				[VC: Element Valid]

Αυτή η προδιαγραφή δεν περιορίζει τη σημασιολογία αλλά χρησιμοποιεί ονόματα των στοιχείων και των χαρακτηριστικών εκτός από τα ονόματα που ξεκινούν ως εξής:

(('X!' 'x') ('M!' 'm') ('L!' 'l')) και είναι δεσμευμένα για τυποποίηση σε αυτήν ή σε μελλοντικές εκδόσεις αυτής της προδιαγραφής.

Περιορισμός Well-formed: Ταίριασμα Του Τύπου Στοιχείου

Το name στο end-tag ενός στοιχείου πρέπει να ταιριάζει με τον τύπο στοιχείου στο start-tag.

Περιορισμός Valid: Valid Στοιχεία

Ένα στοιχείο είναι έγκυρο όταν υπάρχει δήλωση elementdecl όπου το name να ταιριάζει στον τύπο στοιχείου και να ισχύουν τα ακόλουθα:

1. Η δήλωση είναι EMPTY και το στοιχείο δεν έχει περιεχόμενο(content).

2. Η δήλωση είναι children και μια αλληλουχία στοιχείων παιδιού (child elements) ανήκει στη γλώσσα και παράγεται από μια συνήθη έκφραση στο μοντέλο περιεχομένου, με λευκά κενά (χαρακτήρες που να ταιριάζουν στο s) ανάμεσα σε κάθε ζεύγος στοιχείων παιδιού.

3. Η δήλωση είναι mixed και το περιεχόμενο περιέχει χαρακτήρες δεδομένων και στοιχεία παιδιού των οποίων οι τύποι ταιριάζουν με τα ονόματα στο μοντέλο περιεχομένου.

4. Η δήλωση είναι ANY και οι τύποι όλων των στοιχείων παιδιού έχουν δηλωθεί.

3.1 Start-tags, End-tags και Empty-element tags.

Το ξεκίνημα κάθε μη κενού XML στοιχείου γίνεται με ένα **start-tag**.

Start-tag

[40] STag	::=	'<' Name (S Attribute)* S? '>'	[WFC: Unique Att Spec]
[41] Attribute	::=	Name Eq AttValue	[VC: Attribute Value Type]
			[WFC: No External Entity References]
			[WFC: No < in Attribute Values]

Το name στα start και end tags δίνει τον **τύπο** του στοιχείου. Τα ζεύγη name-attvalue αναφέρονται ως **προδιαγραφές χαρακτηριστικών** του στοιχείου, με το name σε κάθε ζεύγος να αναφέρεται στο **χαρακτηριστικό όνομα** και το περιεχόμενο του attvalue στο **χαρακτηριστικό τιμή**.

Περιορισμός Well-formed: Μοναδικό Att Spec

Κανένα χαρακτηριστικό όνομα δεν μπορεί να εμφανιστεί περισσότερη από μια φορά στο ίδιο start-tag ή empty-element tag.

Περιορισμός Valid: Χαρακτηριστικό τύπου τιμής.

Το χαρακτηριστικό πρέπει να δηλωθεί; η τιμή θα πρέπει να ανήκει στον τύπο δήλωσης.

Περιορισμός Well-formed: Καμία Εξωτερική Αναφορά Οντότητας

Τα χαρακτηριστικά τιμής δεν μπορούν να περιέχουν άμεσες ή έμμεσες αναφορές οντοτήτων σε εξωτερικές οντότητες.

Περιορισμός Well-formed: Μη ύπαρξη του συμβόλου < στα χαρακτηριστικά τιμής.

Το κείμενο αντικατάστασης σε οποιαδήποτε οντότητα που αναφέρεται άμεσα ή έμμεσα σε χαρακτηριστικό τιμής (εκτός του "<") δεν πρέπει να περιέχει το σύμβολο <.

Ένα παράδειγμα start-tag είναι το εξής:

```
<termdef id="dt-dog" term="dog">
```

Το τέλος κάθε στοιχείου που ξεκινάει με ένα start-tag πρέπει να δηλώνεται με ένα end-tag το οποίο να περιέχει ένα όνομα που να αντιστοιχεί στον τύπο στοιχείου όπως δίνεται από το start-tag.

End-tag

[42] ETag ::= '</' Name S? '>'

Ένα παράδειγμα από end-tag είναι το εξής:

```
</termdef>
```

Το κείμενο μεταξύ του start-tag και του end-tag ονομάζεται **περιεχόμενο στοιχείου**.

Content of Elements

[43] content ::= (element | CharData | Reference | CDsect | PI | Comment)*

Εάν ένα στοιχείο είναι **κενό**, θα πρέπει να αντιπροσωπεύεται είτε από ένα tag αρχής ακολουθούμενο από ένα tag τέλους είτε από ένα tag κενού στοιχείου (empty element tag). Ένα **tag κενού στοιχείου** παίρνει την εξής μορφή:

Tags for Empty Elements

[44] EmptyElemTag ::= '<' Name (S Attribute)* S? '/>' [WFC: Unique Att Spec]

Τα tag κενού στοιχείου μπορούν να χρησιμοποιηθούν για οποιοδήποτε στοιχείο που δεν έχει περιεχόμενο, είτε είναι δηλωμένο χρησιμοποιώντας τη λέξη EMPTY είτε όχι.

Παραδείγματα κενών στοιχείων:

```
<IMG align="left"
src="http://www.w3.org/Icons/WWW/w3c_home" />
```

```
<br></br>
<br/>
```

3.2 Δηλώσεις Τύπου Στοιχείου (Element Type Declarations)

Η δομή του στοιχείου ενός XML εγγράφου μπορεί, για σκοπούς εγκυρότητας, να είναι περιορισμένη χρησιμοποιώντας τύπους στοιχείων και δηλώσεις λίστας χαρακτηριστικών. Μια δήλωση τύπου στοιχείου περιορίζει το περιεχόμενο του στοιχείου.

Οι δηλώσεις τύπου στοιχείου συχνά καθορίζουν ποιοι τύποι στοιχείων μπορούν να εμφανίζονται ως παιδιά του στοιχείου. Κατά την εκλογή του χρήστη, ο XML επεξεργαστής μπορεί να εισάγει μια ειδοποίηση όταν η δήλωση αναφέρεται σε έναν τύπο στοιχείου για τον οποίο δεν παρέχεται δήλωση, αλλά αυτό δεν είναι λάθος.

Μια δήλωση τύπου στοιχείου παίρνει τη μορφή:

Element Type Declaration

```
[45] elementdecl ::= '<ELEMENT' S Name S contentspec S? '>' [ VC: Unique Element Type Declaration ]
[46] contentspec ::= 'EMPTY' | 'ANY' | Mixed | children
```

Όπου το name δίνει τον τύπο στοιχείου που δηλώνεται.

Περιορισμός Valid: Μοναδική Δήλωση Τύπου Στοιχείου

Κανένας τύπος στοιχείου δεν μπορεί να δηλώνεται περισσότερες από μια φορές.

Παραδείγματα δηλώσεων τύπων στοιχείων:

<ELEMENT	br	EMPTY>
<ELEMENT	p	(#PCDATA emph)*>
<ELEMENT	%name.para;	%content.para;>
<ELEMENT container	ANY>	

3.2.1 Περιεχόμενο Στοιχείου (Element Content)

Ένας τύπος στοιχείου έχει ένα περιεχόμενο στοιχείου, όταν τα στοιχεία αυτού του τύπου περιέχουν υποχρεωτικά στοιχεία παιδιών (και όχι χαρακτήρες δεδομένων), χωρισμένα μεταξύ τους με το λευκό κενό. Σ' αυτήν την περίπτωση, ο περιορισμός περιλαμβάνει ένα μοντέλο περιεχομένου, μια απλή γραμματική που ελέγχει τους επιτρεπτούς τύπους των στοιχείων παιδιών και την σειρά στην οποία επιτρέπονται να εμφανίζονται. Η γραμματική βασίζεται σε μόρια περιεχομένου(cps), τα οποία αποτελούνται από ονόματα, λίστες επιλογής των μορίων περιεχομένου ή λίστες αλληλουχίας των μορίων περιεχομένου:

Element-content Models

```
[47] children ::= (choice | seq) ('?' | '*' | '+')?
[48] cp ::= (Name | choice | seq) ('?' | '*' | '+')?
[49] choice ::= '(' S? cp ( S? '|' S? cp)* S? ')' [ VC: Proper Group/PE Nesting ]
[50] seq ::= '(' S? cp ( S? ',' S? cp)* S? ')' [ VC: Proper Group/PE Nesting ]
```

Όπου κάθε Name είναι ο τύπος του στοιχείου που μπορεί να εμφανιστεί ως παιδί. Οποιοδήποτε μόριο περιεχομένου σε μια λίστα επιλογής μπορεί να εμφανιστεί στο περιεχόμενο στοιχείου στο μέρος όπου εμφανίζεται η λίστα επιλογής στην γραμματική. Τα μόρια περιεχομένου που υπάρχουν σε μια λίστα αλληλουχίας εμφανίζονται στο περιεχόμενο στοιχείου στην θέση που δίνεται από τη λίστα. Ο προαιρετικός χαρακτήρας που ακολουθεί ένα όνομα ή μια λίστα καθορίζει τότε το στοιχείο ή τα μόρια περιεχομένου μπορούν να εμφανίζονται στη λίστα μια ή περισσότερες φορές (+), καμία ή περισσότερες (*), καμία ή μια φορά (?). Η απουσία τέτοιου τελεστή σημαίνει ότι το στοιχείο ή τα μόρια περιεχομένου πρέπει να εμφανίζονται ακριβώς μια φορά. Αυτή η συντακτική και η έννοια είναι παρόμοια με αυτά που χρησιμοποιούνται στην παραγωγή αυτής της προδιαγραφής.

Το περιεχόμενο ενός στοιχείου ταιριάζει στο περιεχόμενο μοντέλου μόνον εάν είναι δυνατόν να δημιουργηθεί ένας διάδρομος μέσω του περιεχομένου μοντέλου, που να υπακούει στην αλληλουχία, στην επιλογή στους επαναληπτικούς τελεστές και να ταιριάζουν σε κάθε στοιχείο στο περιεχόμενο σε αντίθεση με έναν τύπο στοιχείου στο περιεχόμενο μοντέλου. Επιστρέφει σφάλμα εάν το στοιχείο στο έγγραφο συμβεί να ταιριάζει περισσότερες από μια φορές στον τύπο στοιχείου στο περιεχόμενο μοντέλου.

Περιορισμός Valid: Proper Group/PE Nesting

Το κείμενο αντικατάστασης των παραμέτρων οντοτήτων πρέπει να φωλιάζεται κατάλληλα με παρενθετικά σύνολα. Αυτό σημαίνει ότι, οποιεσδήποτε παρενθέσεις ανοίγματος ή κλεισίματος σε μια choice, seg ή mixed κατασκευή που περιέχονται σε κείμενο αντικατάστασης για παραμέτρους οντοτήτων, και οι δυο πρέπει να περιέχονται στο ίδιο κείμενο αντικατάστασης. Εάν μια αναφορά μιας παραμέτρου οντότητας εμφανίζεται σε μια choice, seg ή mixed κατασκευή το κείμενο αντικατάστασής του δεν πρέπει να είναι κενό και ούτε μπορεί να είναι σύνδεσμος ο πρώτος ή ο δεύτερος μη κενός χαρακτήρας του συνδέσμου αντικατάστασης.

Παραδείγματα μοντέλων των στοιχείων περιεχομένου:

<!ELEMENT	spec	(front,	body,	back?)>
<!ELEMENT	div1	(head,	(p list note)*,	div2*)>
<!ELEMENT	dictionary-body	(%div.mix; %dict.mix:)*>		

3.2.2 Μικτό Περιεχόμενο (Mixed Content)

Ένας τύπος στοιχείου έχει μικτό περιεχόμενο όταν τα στοιχεία αυτού του τύπου περιλαμβάνουν χαρακτήρες δεδομένων σε συνδυασμό με στοιχεία παιδιών. Σ' αυτήν την περίπτωση, οι τύποι των στοιχείων παιδιών μπορεί να είναι περιορισμένοι αλλά όχι η διάταξή τους ή ο αριθμός των εμφανίσεών τους:

Mixed-content Declaration

```
[51] Mixed ::= '(' S? '#PCDATA' (S? '|' S? Name)* S? ')'
                | '(' S? '#PCDATA' S? ')'
                [ VC: Proper Group/PE Nesting ]
                [ VC: No Duplicate Types ]
```

όπου τα names δίνουν τον τύπο των στοιχείων που μπορούν να εμφανίζονται ως παιδιά.

Περιορισμός Valid: Δεν υπάρχουν Διπλότυποι Τύποι.

Το ίδιο όνομα δεν μπορεί να εμφανίζεται περισσότερες από μια φορές σε μια μοναδική δήλωση μικτού περιεχομένου.

Παραδείγματα δήλωσης μικτού περιεχομένου:

<!ELEMENT	p	(#PCDATA a ul b i em)*>
<!ELEMENT	p	(#PCDATA %font; %phrase; %special; %form:)* >
<!ELEMENT	b	(#PCDATA)>

3.3 Δήλωση Λίστας Χαρακτηριστικών (Attribute-List Declarations)

Τα χαρακτηριστικά συνδέουν τα ζεύγη ονόματος - τιμής με τα στοιχεία. Οι προδιαγραφές των χαρακτηριστικών εμφανίζονται μονάχα μέσα στα start-tags ή στα empty element tags. Οι δηλώσεις της Λίστας Χαρακτηριστικών μπορούν να χρησιμοποιηθούν:

- Για να καθορίσουν ένα σύνολο χαρακτηριστικών που αναφέρονται σε ένα δοσμένο τύπο στοιχείων.
- Για να καθορίσουν περιορισμούς τύπων για αυτά τα χαρακτηριστικά.
- Για να παρέχουν default τιμές στα χαρακτηριστικά.

Οι δηλώσεις της Λίστας Χαρακτηριστικών καθορίζουν το όνομα, τον τύπο δεδομένων, και την default τιμή του καθενός χαρακτηριστικού που συνδέεται με έναν δοσμένο τύπο στοιχείων:

Attribute-list Declaration

```
[52] AttlistDecl ::= '<!ATTLIST' S Name AttDef* S? '>'
[53] AttDef ::= S Name S AttType S DefaultDecl
```

Το name στον κανόνα AttlistDecl είναι ο τύπος του στοιχείου. Εάν για έναν τύπο στοιχείων που δεν είναι δηλωμένος ο ίδιος δηλώσουμε κάποια χαρακτηριστικά, ο XML επεξεργαστής θα βγάλει μήνυμα προειδοποίησης αλλά όχι λάθους. Το name στον κανόνα AttDef είναι το όνομα του χαρακτηριστικού.

Εάν παρέχονται περισσότερα του ενός AttlistDecl για ένα δοσμένο τύπο στοιχείων, τα περιεχόμενα όλων αυτών συγχωνεύονται. Όταν δίνονται περισσότερες από μια ερμηνείες στο ίδιο χαρακτηριστικό ενός δοσμένου τύπου στοιχείων, η πρώτη δήλωση είναι δεσμεύσιμη και οι επόμενες αγνοούνται. Οι συγγραφείς του DTD μπορεί να επιλέξουν να παρέχουν τουλάχιστον μια δήλωση λίστας χαρακτηριστικών για ένα δοσμένο τύπο στοιχείων, τουλάχιστον μια ερμηνεία χαρακτηριστικού για ένα δοσμένο χαρακτηριστικό ονόματος και τουλάχιστον μια ερμηνεία χαρακτηριστικού για κάθε δήλωση λίστας χαρακτηριστικών. Όταν περισσότερες από μια δηλώσεις λίστας χαρακτηριστικών παρέχονται για ένα δοσμένο τύπο στοιχείων ή όταν περισσότερες από μια ερμηνείες χαρακτηριστικών παρέχονται σε ένα δοσμένο χαρακτηριστικό, ο XML επεξεργαστής εισάγει προειδοποίηση και όχι μήνυμα λάθους.

3.3.1 Τύποι Χαρακτηριστικών (Attribute Types)

Οι XML τύποι χαρακτηριστικών είναι τρεις:

1. Ο τύπος `string`, ο οποίος μπορεί να πάρει ως τιμή οποιοδήποτε αληθινό `string`.
2. Ένα σύνολο συμβολικών τύπων, οι οποίοι έχουν ποικίλους λεξιλογικούς και εννοιολογικούς περιορισμούς, τους εξής:

Attribute Types

[54]	<code>AttType</code>	::=	<code>StringType</code> <code>TokenizedType</code> <code>EnumeratedType</code>	
[55]	<code>StringType</code>	::=	<code>'CDATA'</code>	
[56]	<code>TokenizedType</code>	::=	<code>'ID'</code>	[VC: ID]
				[VC: One ID per Element Type]
				[VC: ID Attribute Default]
			<code>'IDREF'</code>	[VC: IDREF]
			<code>'IDREFS'</code>	[VC: IDREF]
			<code>'ENTITY'</code>	[VC: Entity Name]
			<code>'ENTITIES'</code>	[VC: Entity Name]
			<code>'NMTOKEN'</code>	[VC: Name Token]
			<code>'NMTOKENS'</code>	[VC: Name Token]

Περιορισμός Valid:ID

Οι τιμές των τύπων `ID` πρέπει να ταιριάζουν στο προϊόν `name`. Το `name` δεν πρέπει να εμφανίζεται περισσότερες από μια φορές στο XML έγγραφο ως τιμή αυτού του τύπου, π.χ. οι `ID` τιμές θα πρέπει να προσδιορίζουν τα στοιχεία που τις παράγουν.

Περιορισμός Valid: Ένα ID για κάθε τύπο στοιχείων

Κανένας τύπος στοιχείων δεν μπορεί να έχει περισσότερα από ένα καθορισμένα `ID` χαρακτηριστικά.

Περιορισμός Valid: ID Χαρακτηριστικό Μηδενικής Απόδοσης Τιμής

Ένα `ID` χαρακτηριστικό παίρνει ως default τιμές τις εξής: `#implied` / `#required`

Περιορισμός Valid: IDREF

Οι τιμές του τύπου `IDREF` πρέπει να ταιριάζουν στο προϊόν `name`, και οι τιμές του τύπου `IDREFS` πρέπει να ταιριάζουν στο προϊόν `names`. Κάθε `name` πρέπει να αντιστοιχεί στην αξία ενός `ID` χαρακτηριστικού σε κάποιο στοιχείο σε ένα XML έγγραφο, π.χ. οι τιμές `IDREF` θα πρέπει να αντιστοιχούν στην αξία κάποιων `ID` χαρακτηριστικών.

Περιορισμός Valid: Όνομα Οντότητας

Οι τιμές του τύπου `ENTITY` πρέπει να ταιριάζουν στο προϊόν `name`, και οι τιμές του τύπου `ENTITIES` πρέπει να ταιριάζουν στο προϊόν `names`. Κάθε `name` πρέπει να ταιριάζει με το όνομα μιας μη αναλυμένης οντότητας που δηλώνεται στο DTD.

Περιορισμός Valid: Σύμβολο Ονόματος

Οι τιμές του τύπου `NMTOKEN` πρέπει να ταιριάζουν στο προϊόν `Nmtoken`, και οι τιμές του τύπου `NMTOKENS` πρέπει να ταιριάζουν στο προϊόν `Nmtokens`.

3. Οι αριθμημένοι τύποι

Τα **αριθμημένα χαρακτηριστικά** μπορούν να πάρουν μια τιμή από τη λίστα που παρέχεται στην δήλωση. Υπάρχουν δυο είδη αριθμημένων τύπων:

Enumerated Attribute Types

[57]	<code>EnumeratedType</code>	::=	<code>NotationType</code> <code>Enumeration</code>	
[58]	<code>NotationType</code>	::=	<code>'NOTATION'</code> S <code>'('</code> S? <code>Name</code> (S? <code>' '</code> S? <code>Name</code>)* S? <code>)'</code>	[VC: Notation Attributes]
[59]	<code>Enumeration</code>	::=	<code>'('</code> S? <code>Nmtoken</code> (S? <code>' '</code> S? <code>Nmtoken</code>)* S? <code>)'</code>	[VC: Enumeration]

Το χαρακτηριστικό `NOTATION` προσδιορίζει ένα σχόλιο, δηλώνεται στο DTD με κοινούς προσδιοριστές για να χρησιμοποιείται στη μετάφραση του στοιχείου στο οποίο ανήκει το χαρακτηριστικό.

Περιορισμός Valid: Χαρακτηριστικά Σχολίου

Οι τιμές αυτού του τύπου θα πρέπει να ταιριάζουν σε ένα από τα ονόματα σχολίων που περιλαμβάνει η δήλωση: όλα τα ονόματα σχολίων στην δήλωση θα πρέπει να δηλωθούν.

Περιορισμός Valid: Απαρίθμηση

Οι τιμές αυτού του τύπου πρέπει να ταιριάζουν σε ένα από τα σύμβολα Nmtoken στην δήλωση.

Το ίδιο Nmtoken δεν θα συμβεί περισσότερες από μια φορές στον τύπο αριθμημένων χαρακτηριστικών ενός μοναδικού τύπου στοιχείου.

3.3.2 Χαρακτηριστικά Μηδενικής Τιμής

Μια δήλωση χαρακτηριστικών παρέχει πληροφορίες για το αν είναι απαραίτητη η παρουσία του χαρακτηριστικού, και αν όχι πώς αντιδρά ο XML επεξεργαστής αν δηλωμένα χαρακτηριστικά απουσιάζουν σε ένα κείμενο.

Attribute Defaults

[60]	DefaultDecl	::=	'#REQUIRED' '#IMPLIED'		
			((' #FIXED' S)? AttValue)	[VC: Required Attribute]
				[VC: Attribute Default Legal]
				[WFC: No < in Attribute Values]
				[VC: Fixed Attribute Default]

Σε μια δήλωση χαρακτηριστικών το #required σημαίνει ότι το χαρακτηριστικό πρέπει να παρέχεται πάντοτε ενώ το #implied ότι δεν παρέχεται καμία default τιμή. Αν η δήλωση δεν είναι ούτε #required ούτε #implied τότε η τιμή AttValue περιέχει την δηλωμένη default τιμή. Η λέξη-κλειδί #fixed δηλώνει ότι το χαρακτηριστικό πρέπει να έχει πάντοτε την default τιμή. Αν μια default τιμή είναι δηλωμένη, όταν ο XML επεξεργαστής αντιμετωπίζει ένα χαρακτηριστικό που έχει παραληφθεί, συμπεριφέρεται σαν να ήταν παρόν το χαρακτηριστικό με τη δηλωμένη default τιμή.

Περιορισμός Valid: Απαιτούμενο Χαρακτηριστικό

Αν η default δήλωση είναι η λέξη-κλειδί #required τότε το χαρακτηριστικό πρέπει να καθοριστεί για όλα τα στοιχεία του τύπου στην δήλωση της λίστας χαρακτηριστικών.

Περιορισμός Valid: Νόμιμο χαρακτηριστικό Default

Η δηλωμένη default τιμή πρέπει να συναντήσει τους λεξιλογικούς περιορισμούς του δηλωμένου τύπου χαρακτηριστικών.

Περιορισμός Valid: Αμετάβλητη τιμή Default

Αν ένα χαρακτηριστικό έχει μια default τιμή δηλωμένη με τη λέξη-κλειδί #fixed, τα παραδείγματα αυτού του χαρακτηριστικού θα πρέπει να ταιριάζουν με την default τιμή.

Παραδείγματα της δήλωσης λίστας χαρακτηριστικών:

```
<!ATTLIST termdef
    id ID #REQUIRED
    name CDATA #IMPLIED>
<!ATTLIST list
    type (bullets|ordered|glossary) "ordered">
<!ATTLIST form
    method CDATA #FIXED "POST">
```

3.3.3 Κανονικοποίηση Απόδοσης Τιμής (Attribute-Value Normalization)

Πριν περαστεί η τιμή του χαρακτηριστικού στην εφαρμογή ή πριν ελεγχθεί για αξιοπιστία, ο XML επεξεργαστής πρέπει να την κανονικοποιήσει ως εξής:

- Μια αναφορά χαρακτηριστικού παράγεται προσαρτώντας τον δηλωμένο χαρακτήρα στην τιμή του χαρακτηριστικού.
- Μια αναφορά οντότητας παράγεται αν επεξεργάζεται περιοδικά το κείμενο αντικατάστασης της οντότητας.
- Ένας λευκός χαρακτήρας (#x20, #xD, #xA, #x9) παράγεται προσαρτώντας το #x20 στην κανονικοποιημένη τιμή, εκτός από το γεγονός ότι μονάχα ένα #x20 προσαρτάται σε μια αλληλουχία "#xD, #xA" που είναι μέρος μιας εξωτερικής αναλυμένης οντότητας ή της αληθινής οντότητας μιας εσωτερικής αναλυμένης οντότητας.
- Άλλοι χαρακτήρες παράγονται προσαρτώντας τους στην κανονικοποιημένη τιμή.

Αν η δηλωμένη τιμή δεν είναι CDATA, ο XML επεξεργαστής πρέπει να επεξεργαστεί μετέπειτα την κανονικοποιημένη τιμή του χαρακτηριστικού απορρίπτοντας οποιουδήποτε κύριους και δευτερεύοντες χαρακτήρες κενού αντικαθιστώντας αλληλουχίες κενών χαρακτήρων με έναν μοναδικό κενό χαρακτήρα.

Τα χαρακτηριστικά για τα οποία δεν υπάρχει δήλωση πρέπει να συμπεριφέρονται από έναν μη έγκυρο αναλυτή ως δηλωμένα CDATA.

3.4 Υποθετικοί Τομείς (Conditional Sections)

Οι **υποθετικοί τομείς** είναι τμήματα της δήλωσης τύπου εγγράφου εξωτερικού υποσυνόλου τα οποία περιέχονται ή εξέρχονται από τη λογική δομή ενός DTD βασιζόμενου στη λέξη-κλειδί που τα καθορίζει.

Conditional Section

[61]	conditionalSect	::=	includeSect ignoreSect
[62]	includeSect	::=	'<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
[63]	ignoreSect	::=	'<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>'
[64]	ignoreSectContents	::=	Ignore ('<![' ignoreSectContents ']]>' Ignore)*
[65]	Ignore	::=	Char* - (Char* ('<![' ']]>') Char*)

Όπως τα εσωτερικά και τα εξωτερικά DTD υποσύνολα, ένας υποθετικός τομέας μπορεί να περιλαμβάνει μια ή περισσότερες ολοκληρωμένες δηλώσεις, σχόλια, οδηγίες επεξεργασίας αναμειγνυόμενα με το «λευκό κενό».

Αν η λέξη-κλειδί του υποθετικού τμήματος είναι INCLUDE, τότε τα περιεχόμενα του υποθετικού τμήματος είναι τμήμα του DTD. Αν η λέξη-κλειδί του υποθετικού τμήματος είναι IGNORE, τότε τα περιεχόμενα του υποθετικού τμήματος δεν είναι τμήμα του DTD. Για αξιόπιστη ανάλυση, τα περιεχόμενα ακόμη και των πιο αγνοημένων υποθετικών τμημάτων πρέπει να μελετούνται έτσι ώστε να εντοπίζονται φωλιασμένοι υποθετικοί τομείς και να προσδιορίζεται το τέλος τους. Όταν υπάρχει ένας υποθετικός τομέας με τη λέξη-κλειδί INCLUDE μέσα σε ένα μεγαλύτερο υποθετικό τομέα με τη λέξη-κλειδί IGNORE, τόσο ο εξωτερικός όσο και ο εσωτερικός υποθετικός τομέας αγνοούνται.

Αν η λέξη-κλειδί ενός υποθετικού τομέα είναι μια αναφορά οντότητας παραμέτρων, αυτή πρέπει να αντικαθιστάται από το περιεχόμενό του πριν ο επεξεργαστής αποφασίσει αν περιλάβει ή αν αγνοήσει τον υποθετικό τομέα.

Ένα παράδειγμα:

```
<ENTITY % draft 'INCLUDE' >
<ENTITY % final 'IGNORE' >

<![%draft;[
<ELEMENT book (comments*, title, body, supplements?)>
]]>
<![%final;[
<ELEMENT book (title, body, supplements?)>
]]>
```

4. Φυσικές Δομές (Physical Structures)

Ένα XML έγγραφο αποτελείται από μια ή από πολλές μονάδες αποθήκευσης. Αυτές καλούνται **οντότητες(entities)**: έχουν όλες **περιεχόμενο ((content))** εκτός από την οντότητα εγγράφου) και είναι όλες προσδιοριστές από ένα όνομα (name). Κάθε XML έγγραφο έχει μια οντότητα που καλείται οντότητα εγγράφου, η οποία λειτουργεί ως σημείο αρχής στον XML επεξεργαστή και μπορεί να περιέχει ολόκληρο το έγγραφο.

Οι οντότητες είναι είτε αναλυμένες είτε όχι. Τα περιεχόμενα μιας **αναλυμένης οντότητας** αναφέρονται ως κείμενο αντικατάστασης· αυτό το περιεχόμενο θεωρείται αναπόσπαστο μέρος του εγγράφου.

Μια **μη αναλυμένη οντότητα** είναι μια πηγή της οποίας τα περιεχόμενα μπορεί να είναι κείμενο ή όχι, και αν είναι κείμενο μπορεί να μην είναι XML. Κάθε μη αναλυμένη οντότητα είναι συνδεδεμένη με ένα σχόλιο, το οποίο προσδιορίζεται από το όνομά της. Πίσω από την απαίτηση ότι ένας επεξεργαστής διαθέτει τους προσδιοριστές για την οντότητα και το σχόλιο στην εφαρμογή, η XML δεν θέτει καθόλου περιορισμούς στα σχόλια των μη αναλυμένων οντοτήτων.

Οι αναλυμένες οντότητες υλοποιούνται από ένα όνομα χρησιμοποιώντας αναφορές οντοτήτων, ενώ οι μη αναλυμένες από ένα όνομα που δίνεται από τις τιμές των χαρακτηριστικών ENTITY ή ENTITIES.

Οι **γενικές οντότητες** είναι οντότητες που χρησιμοποιούνται μέσα στο περιεχόμενο του κειμένου. Σ' αυτήν την προδιαγραφή, οι γενικές οντότητες αναφέρονται συχνά με τον αναρμόδιο όρο entity όταν δεν οδηγεί σε κάποια αμφιβολία. Οι οντότητες παραμέτρων είναι αναλυμένες οντότητες για χρήση μέσα σε ένα DTD. Αυτοί οι δυο τύποι οντοτήτων χρησιμοποιούν αναφορές διαφορετικού τύπου και αναγνωρίζονται σε διαφορετικό περιβάλλον. Ακόμη απασχολούν διαφορετικά namespaces· μια οντότητα παραμέτρων και μια γενική οντότητα με το ίδιο όνομα είναι δυο ευδιάκριτες οντότητες.

4.1 Αναφορές Χαρακτήρων και Οντοτήτων (Character and Entity References)

Μια **αναφορά χαρακτήρα** αναφέρεται σε έναν συγκεκριμένο χαρακτήρα του συνόλου χαρακτήρων του ISO/IEC 10646, για παράδειγμα σε ένα όχι άμεσα προσβάσιμο από τις διαθέσιμες συσκευές εισόδου.

Character Reference

```
[66] CharRef ::= '&#'[0-9]+';'
           | '&#x'[0-9a-fA-F]+';' [ WFC: Legal Character ]
```

Περιορισμός Well-formed: Νόμιμος Χαρακτήρας

Οι χαρακτήρες που αναφέρονται χρησιμοποιώντας αναφορές χαρακτήρων πρέπει να ταιριάζουν στο προϊόν για char.

Αν η αναφορά χαρακτήρα ξεκινά με "&#x", τα ψηφία και τα γράμματα στο τέλος ; παρέχουν μια δεκαεξαδική αναπαράσταση του κώδικα χαρακτήρων του ISO/IEC 10646. Αν ξεκινά με "&#", τα ψηφία στο τέλος ; παρέχουν μια δεκαδική αναπαράσταση του κώδικα χαρακτήρων.

Μια **αναφορά οντότητας** αναφέρεται στο περιεχόμενο μιας ονομαστικής οντότητας. Οι αναφορές σε αναλυμένες γενικές οντότητες χρησιμοποιούν ως χαρακτήρες αρχής και τέλους τους (&) και (;). Οι **αναφορές παραμέτρων οντοτήτων** χρησιμοποιούν τους (%) και (;).

Entity Reference

```
[67] Reference ::= EntityRef | CharRef
[68] EntityRef ::= '&' Name ';' [ WFC: Entity Declared ]
                                   [ VC: Entity Declared ]
                                   [ WFC: Parsed Entity ]
                                   [ WFC: No Recursion ]
[69] PReference ::= '%' Name ';' [ VC: Entity Declared ]
                                   [ WFC: No Recursion ]
                                   [ WFC: In DTD ]
```

Περιορισμός Well-formed: Δηλωμένες Οντότητες

Σε ένα έγγραφο με κανένα DTD, ένα έγγραφο με μόνο ένα εσωτερικό υποσύνολο που δεν περιέχει καμία αναφορά παραμέτρων οντοτήτων, ή σε ένα έγγραφο με "standalone = 'yes' ", το **name** που δίνεται στην αναφορά οντοτήτων πρέπει να ταιριάζει με αυτό σε μια **δήλωση οντοτήτων**, εκτός από τα well-formed έγγραφα που δεν υποχρεούνται να δηλώνουν μια από τις παρακάτω οντότητες: amp, lt, gt, apos, quot. Η δήλωση των παραμέτρων οντοτήτων πρέπει να προηγείται οποιασδήποτε αναφοράς σε αυτές. Παράλληλα, η δήλωση μιας γενικής οντότητας πρέπει να προηγείται οποιασδήποτε αναφοράς σε αυτήν, η οποία εμφανίζεται με μια default τιμή σε μια δήλωση λίστας χαρακτηριστικών. Αν οι οντότητες δηλώνονται σε εξωτερικά υποσύνολα ή σε εξωτερικές παραμέτρους οντοτήτων ένας μη έγκυρος επεξεργαστής δεν είναι υποχρεωμένος να διαβάζει και να επεξεργάζεται αυτές τις δηλώσεις. Για αυτά τα έγγραφα ο κανόνας ότι μια οντότητα πρέπει να δηλώνεται είναι ένας well-formed περιορισμός μόνο αν ισχύει standalone = 'yes'.

Περιορισμός Valid: Δηλωμένες Οντότητες

Σε ένα έγγραφο με εξωτερικά υποσύνολα ή εξωτερικές παραμέτρους οντοτήτων με "standalone = 'no' ", το **name** που δίνεται στην αναφορά οντοτήτων πρέπει να ταιριάζει με αυτό σε μια **δήλωση οντοτήτων**. Τα valid έγγραφα πρέπει να δηλώνουν τις οντότητες amp, lt, gt, apos, quot. Η δήλωση των παραμέτρων οντοτήτων πρέπει να προηγείται οποιασδήποτε αναφοράς σε αυτές. Παράλληλα, η δήλωση μιας γενικής οντότητας πρέπει να προηγείται οποιασδήποτε αναφοράς σε αυτήν, η οποία εμφανίζεται με μια default τιμή σε μια δήλωση λίστας χαρακτηριστικών.

Περιορισμός Well-formed: Αναλυμένη Οντότητα

Μια αναφορά οντότητας δεν πρέπει να περιλαμβάνει το όνομα μιας μη αναλυμένης οντότητας. Οι μη αναλυμένες οντότητες πρέπει να αναφέρονται μόνο σε αποδόσεις τιμών δηλωμένες του τύπου ENTITY ή ENTITIES.

Περιορισμός Well-formed: Χωρίς Επιστροφή

Μια αναλυμένη οντότητα δεν πρέπει να περιλαμβάνει περιοδικά επαναλαμβανόμενες αναφορές στον εαυτό της, είτε άμεσα είτε έμμεσα.

Περιορισμός Well-formed: Σε DTD

Οι αναφορές παραμέτρων οντοτήτων μπορούν να εμφανίζονται μονάχα σε DTD.

Παραδείγματα αναφορών χαρακτήρων και οντοτήτων:

Type <key>less-than</key> (<) to save options.

This document was prepared on &docdate; and
is classified &security-level;.

Παράδειγμα αναφοράς παραμέτρου οντοτήτων :

```
<!-- declare the parameter entity "ISOLat2"... -->
<!ENTITY % ISOLat2
SYSTEM "http://www.xml.com/iso/isolat2-xml.entities" >
<!-- ... now reference it. -->
%ISOLat2;
```

4.2 Δηλώσεις Οντοτήτων (Entity Declarations)

Οι οντότητες δηλώνονται ως εξής:

Entity Declaration

[70]	EntityDecl	::=	GEDecl PEDecl
[71]	GEDecl	::=	'<ENTITY' S Name S EntityDef S? '>'
[72]	PEDecl	::=	'<ENTITY' S '%' S Name S PEDef S? '>'
[73]	EntityDef	::=	EntityValue (ExternalID NDataDecl?)
[74]	PEDef	::=	EntityValue ExternalID

Το name προσδιορίζει την οντότητα σε μια αναφορά οντοτήτων, ή στην περίπτωση μιας μη αναλυμένης οντότητας, στην τιμή του χαρακτηριστικού ENTITY ή ENTITIES. Αν η ίδια οντότητα είναι δηλωμένη περισσότερες από μια φορές, η πρώτη δήλωση που αντιμετωπίζεται είναι δεσμευτική. Σύμφωνα με επιλογή του χρήστη, ο XML επεξεργαστής μπορεί να εισάγει μια προειδοποίηση αν οι οντότητες δηλώνονται πολλαπλές φορές.

4.2.1 Εσωτερικές Οντότητες (Internal Entities)

Αν ο ορισμός της οντότητας είναι ένα EntityValue, η καθορισμένη οντότητα καλείται εσωτερική οντότητα. Δεν υπάρχει ξεχωριστό αντικείμενο φυσικής αποθήκευσης και το περιεχόμενο της οντότητας δίνεται στην δήλωση.

Η εσωτερική οντότητα είναι μια αναλυμένη οντότητα.

Παράδειγμα δήλωσης εσωτερικής οντότητας:

```
<ENTITY Pub-Status "This is a pre-release of the
specification.">
```

4.2.2 Εξωτερικές Οντότητες (External Entities)

Οι εξωτερικές οντότητες δηλώνονται ως εξής:

External Entity Declaration

[75]	ExternalID	::=	'SYSTEM' S SystemLiteral 'PUBLIC' S PubidLiteral S SystemLiteral
[76]	NdataDecl	::=	S 'NDATA' S Name [VC: Notation Declared]

Αν υπάρχει το NdataDecl, αυτή είναι μια γενική μη αναλυμένη οντότητα, διαφορετικά είναι αναλυμένη οντότητα.

Περιορισμός Valid: Δήλωση Σχολίου

Το name πρέπει να ταιριάζει με το δηλωμένο όνομα του σχολίου.

SystemLiteral καλείται ο αναγνωριστής οντοτήτων του συστήματος. Είναι μια URI που χρησιμοποιείται για να ανακτήσει την οντότητα. Ο χαρακτήρας (#) και ο προσδιοριστής τεμαχίου χρησιμοποιούνται συχνά με τα URI και αποτελούν, όχι όμως τυπικά, μέρος τους. Ο XML επεξεργαστής μπορεί να επισημάνει λάθος εάν ο προσδιοριστής τεμαχίου αποτελεί μέρος του αναγνωριστή συστήματος. Τα αναφορικά URI είναι σχετικά με τη δήλωση της πηγής μέσα στην οποία πραγματοποιείται η δήλωση οντοτήτων. Τότε τα URI είναι σχετικά με την οντότητα των εγγράφων, με την οντότητα που περιέχει εξωτερικά DTD υποσύνολα ή με άλλες εξωτερικές παραμέτρους οντοτήτων.

Ένας XML επεξεργαστής μπορεί να χειριστεί έναν χαρακτήρα που δεν είναι ASCII μέσα σε ένα URI αναπαριστώντας τον χαρακτήρα UTF-8 ως ένα ή περισσότερα bytes, και μετά αποφεύγοντας αυτά τα bytes με έναν URI μηχανισμό escaping (π.χ. μετατρέποντας κάθε byte σε %HH, όπου HH είναι το δεκαεξαδικό σχόλιο της τιμής του byte).

Επιπρόσθετα σε έναν αναγνωριστή συστήματος, ο εξωτερικός αναγνωριστής περιλαμβάνει και έναν **κοινό αναγνωριστή**. Ο επεξεργαστής προσπαθώντας να ανακτήσει τα περιεχόμενα της οντότητας μπορεί να χρησιμοποιήσει τον κοινό αναγνωριστή για να προσπαθήσει να παράγει ένα εναλλακτικό URI. Αν ωστόσο δεν μπορέσει να το κάνει αυτό, μπορεί να χρησιμοποιήσει τα προκαθορισμένα URI που υπάρχουν στο αληθινό σύστημα. Πριν γίνει η προσπάθεια για

οποιοδήποτε συνταίριασμα, όλα τα String του «λευκού κενού» στον κοινό αναγνωριστή πρέπει να κανονικοποιηθούν σε χαρακτήρες μονού κενού (#x20), καθώς και τα πρώτα και τα τελευταία κενά να απομακρυνθούν.

Παραδείγματα δηλώσεων εξωτερικών οντοτήτων:

```
<!ENTITY open-hatch
SYSTEM "http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY open-hatch
PUBLIC "-//Textuality//TEXT Standard open-hatch boilerplate//EN"
"http://www.textuality.com/boilerplate/OpenHatch.xml">
<!ENTITY hatch-pic
SYSTEM "../grafix/OpenHatch.gif"
NDATA gif >
```

4.3 Αναλυμένες Οντότητες (Parsed Entities)

4.3.1 Η Δήλωση Κειμένου(The Text Declaration)

Οι εξωτερικές αναλυμένες οντότητες μπορούν να ξεκινήσουν με μια δήλωση κειμένου.

Text Declaration

```
[77] TextDecl ::= '<?xml' VersionInfo? EncodingDecl S? '?>'
```

Η δήλωση κειμένου πρέπει να εξασφαλιστεί αληθινά και όχι με αναφορά σε μια αναλυμένη οντότητα. Η δήλωση κειμένου εμφανίζεται μονάχα στην αρχή μιας εξωτερικά αναλυμένης οντότητας.

4.3.2 Well-formed Αναλυμένες Οντότητες (Well- Formed Parsed Entities)

Η οντότητα του εγγράφου είναι well-formed αν ταιριάζει στο προϊόν που καλείται document. Μια εξωτερικά αναλυμένη γενική οντότητα είναι well-formed αν ταιριάζει στο προϊόν που καλείται extParsedEnt. Μια εξωτερική παράμετρος οντοτήτων είναι well-formed αν ταιριάζει στο προϊόν που καλείται extPE.

Well-Formed External Parsed Entity

```
[78] extParsedEnt ::= TextDecl? content
```

```
[79] extPE ::= TextDecl? extSubsetDecl
```

Μια εσωτερικά αναλυμένη γενική οντότητα είναι well-formed αν το κείμενο αντικατάστασής της ταιριάζει με το προϊόν που ονομάζεται content. Όλες οι εσωτερικές παράμετροι οντοτήτων είναι εξ' ορισμού well-formed.

Η συνέπεια της καλής μορφοποίησης στις οντότητες είναι ότι η λογική και η φυσική δομή σε ένα XML έγγραφο είναι κατάλληλα φωλιασμένα κανέναν χαρακτήρα αρχής ή τέλους ή κενού, στοιχείο, σχόλιο, οδηγία επεξεργασίας, αναφορά χαρακτήρα ή αναφορά οντότητας μπορεί να ξεκινήσει σε μια οντότητα και να καταλήξει σε άλλη.

4.3.3 Κωδικοποίηση Χαρακτήρων στις Οντότητες (Character Encoding in Entities)

Κάθε εξωτερικά αναλυμένη οντότητα μπορεί να χρησιμοποιήσει διαφορετικές κωδικοποιήσεις για τα χαρακτηριστικά της. Όλοι οι XML επεξεργαστές θα πρέπει να είναι ικανοί να διαβάζουν οντότητες και σε UTF-8 και σε UTF-16.

Οι οντότητες που κωδικοποιούνται σε UTF-16 ξεκινούν με το Byte Order Mask όπως περιγράφεται από το ISO/IEC 10646 Annex E και Unicode Appendix B. Αυτή είναι μια υπογραφή κωδικοποίησης και όχι μέρος του markup ή των χαρακτήρων δεδομένων στο XML έγγραφο. Οι XML επεξεργαστές θα πρέπει να είναι ικανοί να χρησιμοποιούν αυτόν το χαρακτήρα για να διαφοροποιούν τα κωδικοποιημένα έγγραφα σε UTF-8 και UTF-16.

Παρ' όλο που οι XML επεξεργαστές απαιτείται να διαβάζουν οντότητες σε UTF-8 και UTF-16 κωδικοποιήσεις, είναι γνωστό ότι υπάρχουν στον κόσμο και άλλες κωδικοποιήσεις και μπορεί να είναι επιθυμητό από τους XML επεξεργαστές να διαβάζουν οντότητες που τις χρησιμοποιούν. Οι αναλυμένες οντότητες που αποθηκεύονται σε μια κωδικοποίηση εκτός των UTF-8 και UTF-16, πρέπει να ξεκινούν με μια δήλωση κειμένου που να περιέχει μια κωδικοποιημένη δήλωση:

Encoding Declaration

```
[80] EncodingDecl ::= S 'encoding' Eq ( '"' EncName '"' | "'"
EncName "'" )
```

```
[81] EncName ::= [A-Za-z] ([A-Za-z0-9._] | '-')* /* Encoding name contains only Latin characters */
```

Στην οντότητα εγγράφου η κωδικοποιημένη δήλωση είναι μέρος της XML δήλωσης. Το EncName είναι το όνομα της κωδικοποίησης που χρησιμοποιείται.

Σε μια κωδικοποιημένη δήλωση, οι τιμές "UTF-8", "UTF-16", "ISO-10646-UCS-2", και "ISO-10646-UCS-4" μπορούν να χρησιμοποιούνται στις ποικίλες κωδικοποιήσεις και μετασχηματισμούς του UNICODE /ISO/IEC 10646, οι τιμές "ISO-8859-1", "ISO-8859-2".....,"ISO-8859-9" " μπορούν να χρησιμοποιούνται στο ISO8859 και οι τιμές "ISO -2022-JP", "Shift_JIS", και "EUC-JP" χρησιμοποιούνται για τις ποικίλες κωδικοποιημένες μορφές του JIS X-0208-1997. Οι XML επεξεργαστές μπορούν να αναγνωρίσουν άλλες κωδικοποιήσεις. Συνιστάται οι κωδικοποιήσεις χαρακτήρων να καταγραφούν στον IANA(Internet Assigned Numbers Authority), και σε σύγκριση με αυτές που βρίσκονται στη λίστα, θα αναφέρονται χρησιμοποιώντας τα ονόματα με τα οποία έχουν καταγραφεί. Αυτά τα ονόματα με τα οποία καταγράφονται έχουν οριστεί να είναι case-insensitive και οι επεξεργαστές θα λειτουργούν με αυτόν τον τρόπο.

Στην απουσία των πληροφοριών που παρέχονται από ένα εξωτερικό πρωτόκολλο μεταφοράς, είναι λάθος για μια οντότητα που περιλαμβάνει μια κωδικοποιημένη δήλωση να παρουσιάζεται στον XML επεξεργαστή με κωδικοποίηση διαφορετική από αυτήν που είχε στην δήλωση καθώς επίσης είναι λάθος για μια κωδικοποιημένη δήλωση να συμβαίνει οπουδήποτε αλλού από την αρχή της εξωτερικής οντότητας και όταν μια οντότητα ούτε ξεκινάει με Byte Order Mark ούτε με κωδικοποιημένη δήλωση που χρησιμοποιεί κώδικα άλλον από τον UTF-8. Από τη στιγμή που ο ASCII είναι ένα υποσύνολο του UTF-8, οι συνήθεις ASCII οντότητες δεν χρειάζονται κωδικοποιημένη δήλωση.Είναι μοιραίο λάθος όταν ο XML επεξεργαστής αντιμετωπίζει οντότητες με κωδικοποίηση που είναι αδύνατον να τις διαχειριστεί.

Παραδείγματα κωδικοποιημένων δηλώσεων:

<?xml	encoding='UTF-8'>
<?xml encoding='EUC-JP'>	

4.4 XML Processor Treatment of Entities and References.

Ο πίνακας παρακάτω συγκεντρώνει το γενικό πλαίσιο στο οποίο παραπομπές σε χαρακτήρα, παραπομπές σε οντότητες, και η επίκληση των unparsed οντοτήτων μπορούν να εμφανιστεί και την απαιτούμενη συμπεριφορά ενός XML processor σε κάθε περίπτωση. Οι ετικέτες στην ακροαριστερή στήλη περιγράφει το γενικό πλαίσιο της αναγνώρισης:

Παραπομπή σε περιεχόμενο: σαν παραπομπή οπουδήποτε μετά την ετικέτα-αρχής και πριν την ετικέτα-τέλους σε ένα στοιχείο, αντιστοιχεί σε ένα όχι οριακό περιεχόμενο content.

Παραπομπή σε τιμές χαρακτηριστικών: σαν παραπομπή διάμεσου είτε τιμών ενός χαρακτηριστικού σε μια ετικέτα-αρχής, ή default τιμή σε μια δήλωση χαρακτηριστικού, αντιστοιχεί σε όχι οριακό AttValue.

Εμφάνιση σαν τιμή χαρακτηριστικού: σαν Name, όχι παραπομπή, εμφανιζόμενη είτε σαν τιμή ενός χαρακτηριστικού που δηλώθηκε σαν τύπος ENTITY (οντότητα), ή σαν ένα από τα space-separated δείγματα στην τιμή του χαρακτηριστικού που είχε δηλωθεί σαν τύπου ENTITIES.

Παραπομπή σε τιμή οντότητας (Entity Value): σαν παραπομπή σε μια παράμετρο ή εσωτερική επακριβή τιμή οντότητας σε οντότητα σε μια δήλωση οντότητας, αντιστοιχεί σε όχι οριακό EntityValue.

Παραπομπή σε DTD: Σαν παραπομπή σε είτε εσωτερικό ή εξωτερικό subset του DTD, αλλά έξω από EntityValue ή Attvalue.

	Παράμετρος	Τύπος Οντότητας Εσωτερικά γενικά	Εξωτερικά γενικά	Parsed	Un- parsed	Χαρακτήρας
Παραπομπή σε περιεχόμενο	Όχι αναγνωρισμένα	Συμπεριλαμβανομένα	Συμπεριλαμβανομένα αν επικυρωθούν		Απαγορευμένα	Συμπεριλαμβανομένα
Παραπομπή σε τιμές χαρακτηριστικών	Όχι αναγνωρισμένα	Συμπεριλαμβανομένα επ'ακριβώς	Απαγορευμένα		Απαγορευμένα	Συμπεριλαμβανομένα
Εμφάνιση σαν τιμή χαρακτηριστικού	Όχι αναγνωρισμένα	Απαγορευμένα	Απαγορευμένα		Notify	Όχι αναγνωρισμένα
Παραπομπή σε τιμή οντότητας (Entity Value)	Συμπεριλαμβανομένα επ'ακριβώς	Παραπεμπτά	Παραπεμπτά		Απαγορευμένα	Συμπεριλαμβανομένα
Παραπομπή σε DTD	Συμπεριλαμβανομένα σαν PE	Απαγορευμένα	Απαγορευμένα		Απαγορευμένα	Απαγορευμένα

4.4.1 Όχι αναγνωρισμένα.

Εκτός του DTD, ο χαρακτήρας "%" δεν έχει ειδική σημασία, έτσι επιπλέον ο,τι θα μπορούσε να γίνουν παραπομπές σε παραμέτρους οντότητα στην DTD δεν αναγνωρίζεται σαν markup σε content. Παρόμοια, τα ονόματα των unparsed οντοτήτων δεν αναγνωρίζονται εκτός αν εκφράζουν τιμές από σωστά δηλωμένα χαρακτηριστικά.

4.4.2 Συμπεριλαμβανόμενα.

Μια οντότητα **συμπεριλαμβάνεται** όταν το κείμενο αντικαταστήσει του προσκομισθεί και επεξεργασθεί, στη θέση της ίδιας της παραπομπής, σαν να ήταν μέρος του εγγράφου στη τοποθεσία που η παραπομπή αναγνωρίστηκε. Το κείμενο αντικατάστασης μπορεί να περιέχει και δεδομένα χαρακτήρες και markup, το οποίο πρέπει να αναγνωρίζεται με τον συνηθισμένο τρόπο, εκτός από το ότι το κείμενο αντικατάσταση από οντότητες χρησιμοποιείται για να διαφεύγει τους οριοθετείς markup(τις οντότητες amp, it, gt, apo, quot) πάντα το βλέπουμε σαν data. Μια αναφορά σε χαρακτήρα **συμπεριλαμβάνεται** όταν ο υποδεικνυόμενος χαρακτήρας επεξεργάζεται αυτή της παραπομπής της ίδιας.

4.4.3 Συμπεριλαμβανόμενο αν επικυρωθεί (included if Validating).

Όταν ο XML επεξεργαστής αναγνωρίζει μια παραπομπή σε parse οντότητα, προκειμένου να επικυρώσει το έγγραφο, ο επεξεργαστής πρέπει να περιέχει τα κείμενα αντικατάστασης του. Αν η οντότητα είναι εξωτερική, και ο επεξεργαστής δεν σκοπεύει να επικυρώσει το XML έγγραφο, ο επεξεργαστής μπορεί, αλλά δεν είναι απαραίτητο, να περιλαμβάνει τα κείμενα αντικαταστατής της οντότητας. Αν ένα non-validating parser δεν περιλαμβάνει το κείμενα αντικατάστασης του, πρέπει να ενημερώσει την λειτουργία που αναγνώρισε, αλλά δεν διάβασε, την οντότητα.

Αυτός ο κανόνας είναι βασισμένος στην αναγνώριση της αυτόματης συμπερίληψης που παρέχεται από τους SGML και XML μηχανισμούς οντοτήτων κυρίως σχεδιασμένοι για να υποστηρίξει modularity στη συγγραφή, δεν είναι απαραίτητα κατάλληλο για αλλά χαρακτηριστικά, in particular document browsing. Οι Browsers, για παράδειγμα, όταν συναντούν μια εξωτερική παραπομπή σε parsed οντότητα μπορεί να διαλέξουν να παρέχουν οπτικό σημάδι της παρουσία της οντότητας και να το ανακτήσει μόνο σε απαίτηση.

4.4.4 Απαγορευμένα.

Τα παρακάτω είναι απαγορευμένα, και συνιστούν fatal errors:

- Η εμφάνιση μιας παραπομπής σε unparsed οντότητα.
- Η εμφάνιση μιας παραπομπής χαρακτήρα ή γενικής οντότητας στο DTD εκτός αν είναι μέσα σε EntityValue ή AttValue.
- Μια παραπομπή σε μια εξωτερική οντότητα σε τιμή χαρακτηριστικού.

4.4.5 Συμπεριλαμβανόμενου επ'ακριβώς.

Όταν μια παραπομπή σε οντότητα εμφανίζεται σε μια τιμή χαρακτηριστικού, ή παραπομπή σε παράμετρο οντότητα εμφανίζεται σε μια επ'ακριβώς τιμή οντότητας, της οποίας το κείμενο αντικατάστασης είναι σε επεξεργασία αντί της ίδιας της παραπομπής, σαν να ήταν μέρος του εγγράφου στο σημείο που αναγνωριστική η παραπομπή, εκτός από ένα απλό ή διπλό χαρακτήρα παραπομπής στο κείμενο αντικατάσταση, του συμπεριφέρονται πάντα σαν κανονικό data χαρακτήρα και δεν τερματίζει το literal. Τίχ αυτό είναι καλά σχεδιασμένο:

```
<!ENTITY % YN "Yes" >
```

```
<!ENTITY WhatHeSaid "He said &YN;" >
```

ενώ αυτό δεν είναι

```
<!ENTITY EndAttr "27'" >
<element attribute='a-&EndAttr;>
```

Σημειώστε.

Όταν το όνομα μιας unparsed οντότητας εμφανίζεται σαν παρμένο από τιμή χαρακτηριστικού δηλωμένου τύπου ENTITY ή ENTITIES, ένας επικυρωμένος επεξεργαστής πρέπει να ενημερώνει την λειτουργία του συστήματος και του κοινού (αν υπάρχει) identifiers και για τις δυο οντότητες και τις σχετικές σημειώσεις τους.

4.4.6 Bypassed (Παρακαμπτεως).

Όταν εμφανίζεται μια γενική παραπομπή σε οντότητα μέσα σε EntityValue σε μια δήλωση Οντότητας, προσπερνάτε αφήνεται όπως είναι.

4.4.7 Συμπεριλαμβανόμενο σαν ΡΕ.

Όπως και με τις εξωτερικές parsed οντότητα, οι παράμετροι οντότητες χρειάζονται μόνο να συμπεριλαμβάνονται αν επικυρώνονται. Όταν μια παραπομπή σε οντότητα-παράμετρο αναγνωρίζεται στο DTD και συμπεριλαμβάνεται, το κείμενο αντικατάστασης του που μεγεθύνεται από το attachment ενός προηγούμενου και ενός επόμενου space (#x20) χαρακτήρα, η πρόθεση είναι να αναγκαστεί το κείμενο αντικατάστασης της οντότητας παραμέτρων να περιέχει ένα ολόκληρο αριθμό από γραμματικά σύμβολα στο DTD.

4.5 Δομή εσωτερική οντότητας κειμένου αντικατάστασης.

Σχετικά με την μεταχείριση των εσωτερικών οντοτήτων, είναι χρήσιμο να ξεχωρίσουμε δυο μορφές τιμών οντοτήτων. Οι **Literal τιμές οντοτήτων** είναι το αναφερθέν string που είναι παρόν στην δήλωση της οντότητας, συμφωνώντας με ο non-terminal EntityValue. Το **κείμενο αντικατάστασης** είναι το περιεχόμενο της οντότητας μετά την αντικατάσταση των παραπομπών σε χαρακτήρα και παραπομπών σε παράμετρος-οντότητες.

Η literal τιμή οντότητα όπως αυτή δίνεται σε μια δήλωση εσωτερική οντότητας (EntityValue) μπορεί να περιέχει παραπομπές χαρακτήρες, παραμέτρους-οντότητες, γενικές οντότητας. Τέτοιες παραπομπές πρέπει να περιέχονται ολοκληρωτικά μέσα σε τιμή Literal οντότητας. Το πραγματικό κείμενο αντικατάστασης που περιέχεται όπως περιγραφική παραπάνω πρέπει να περιέχει το κείμενο αντικατάστασης κάθε παραμέτρους-οντότητας στο οποίο αναφέρεται, και πρέπει να περιέχει το χαρακτήρα στον οποίο αναφέρεται, στη θέση κάθε παραπομπής χαρακτήρα στη literal τιμή οντότητας, παρόλα αυτά παραπομπές γενικής-οντότητας πρέπει να αφήνεται όπως έχει. Για παράδειγμα δίνονται οι παρακάτω δηλώσεις:

```
<!ENTITY % pub "&#xc9;ditions Gallimard" >
<!ENTITY rights "All rights reserved" >
<!ENTITY book "La Peste: Albert Camus,
&#xA9; 1947 %pub;. &rights;" >
```

τότε το κείμενο αντικατάστασης για την οντότητα «book» είναι:

```
La Peste: Albert Camus,
© 1947 Éditions Gallimard. &rights;
```

Η παραπομπή σε γενική οντότητα "&rights;" θα επεκταθεί όταν η παραπομπή "&book;" εμφανιστεί στο περιεχόμενο του εγγράφου ή μια τιμή χαρακτηριστικού.

4.6 Προκαθορισμένες οντότητες.

Οι παραπομπές σε οντότητες και χαρακτήρες μπορούν και οι δυο να χρησιμοποιηθούν για **διαφυγή** από το "}", το "&" και άλλους οριοθέτες. Ένα σετ από γενικές οντότητες (amp, lt, apos, quot) είναι ειδικές για αυτή την περίπτωση. Παραπομπές σε αριθμητικούς χαρακτήρες μπορούν επίσης να χρησιμοποιηθούν, επεκτείνονται αμέσως μόλις αναγνωριστούν και θεωρούνται σαν χαρακτήρες data, έτσι η αναφορά σε αριθμητικό χαρακτήρα "<" και "&" μπορεί να χρησιμοποιηθεί για διαφυγή του < και του & όταν γίνονται σε χαρακτήρα data.

Όλοι οι XML επεξεργαστές πρέπει να αναγνωρίζουν αυτές τις οντότητες είτε είναι δηλωμένες είτε όχι. Για interoperability, ένα έγκυρο XML έγγραφο πρέπει να δηλώνει αυτές τις οντότητες, όπως κάθε άλλη, πριν τις χρησιμοποιήσει. Αν οι οντότητες σε ερωτήσεις είναι δηλωμένες, πρέπει να δηλωθούν σαν εσωτερικές οντότητες των οποίων το κείμενο αντικατάστασης είναι ο απλός χαρακτήρας που διέφυγε ή η παραπομπή χαρακτήρα στο σε αυτό το χαρακτήρα, όπως φαίνεται παρακάτω.

```
<!ENTITY lt "&#38;#60;">
<!ENTITY gt "&#62;">
```

```
<!ENTITY amp "&#38;#38;#38;">
<!ENTITY apos "&#39;">
<!ENTITY quot "&#34;">
```

Σημειώστε ότι οι χαρακτήρες < και το & στις δηλώσεις του "lt" και "amp" έχουν διαφύγει δίπλα για να συναντήσουν τις απαιτήσεις ώστε η αντικατάσταση οντότητας να είναι καλά μορφοποιημένη.

4.7 Δήλωση αριθμητικής παράστασης.

Οι αριθμητικές παραστάσεις αναγνωρίζουν από το όνομα το Format της unparsed οντότητας, το format των στοιχείων που φέρουν ένα χαρακτηριστικό αριθμητικής παράστασης, ή τη λειτουργία στη οποία αναφέρονται οι οδηγίες επεξεργασίας.

Η δήλωση αριθμητικής παράστασης παρέχει ένα όνομα για την παράσταση, για χρήση στην δήλωση οντότητας, στην λίστας-χαρακτηριστικών και τον καθορισμό χαρακτηριστικών, και ένας εξωτερικός identifier για την αριθμητικοί παράσταση ο οποίος μπορεί να επιτρέπει έναν XML επεξεργαστή ή τις client λειτουργίες του να βρίσκουν μια ικανή λειτουργία βοήθειας επεξεργασίας δεδομένων στην δοσμένη παράσταση.

Notation Declarations

```
[82] NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S?
                        '>'
```

```
[83] PublicID ::= 'PUBLIC' S PubidLiteral
```

Οι XML επεξεργαστές πρέπει να παρέχουν εφαρμογές με το όνομα. και εξωτερικούς identifiers για κάθε αριθμητική παράσταση που αναφέρεται και παραπέμπει σε τιμή χαρακτηριστικού, καθορισμό χαρακτηριστικού, ή δήλωση οντότητας. Μπορεί επιπρόσθετα να αναλύουν τον εξωτερικό identifier στο system identifier, όνομα αρχείου, ή άλλες πληροφορίες που χρειάζεται να επιτρέψουν την λειτουργία, να καλέσουν τον επεξεργαστή για δεδομένα στη παράσταση που περιγραφικά.

4.8 Έγγραφο οντότητα.

Το έγγραφο-οντότητα δουλεύει σαν την ρίζα του δένδρου της οντότητας και σημείο-αρχής για τον επεξεργαστή XML. Αυτός ο καθορισμός δεν ορίζει πως το έγγραφο-οντότητα θα τοποθετηθεί από τον XML επεξεργαστή, αντίθετα με άλλες οντότητας, το έγγραφο-οντότητα δεν έχει όνομα και μπορεί να έχει καλή εμφάνιση στη ροή του input του επεξεργαστή χωρίς προσδιορισμούς.

5. Conformance.

5.1 Επικυρωμένοι και μη-επικυρωμένοι επεξεργαστές.

Conforming XML επεξεργαστή πέφτει σε δυο κλάσεις: επικυρωμένοι και μη επικυρωμένοι.

Επικυρωμένοι και μη επικυρωμένοι επεξεργαστές παρόμοια πρέπει να αναφέρουν παραβιάσεις των περιορισμών (που δόθηκαν από την διευκρίνηση) καλής μορφοποίησης περιεχομένου του εγγράφου οντότητα και κάθε άλλων parsed οντοτήτων που διαβάζουν.

Οι επικυρωμένοι επεξεργαστές πρέπει να αναφέρουν παραβιάσεις των περιορισμών που εκφράστηκαν από τις δηλώσεις στο DTD, και αποτυχίες στην εκπλήρωση περιορισμού που δόθηκε από αυτήν διευκρίνηση. Για να επιτευχθεί αυτή η επικύρωση ο XML επεξεργαστής πρέπει να διαβάζει και να επεξεργάζεται όλο το DTD και όλες τις εξωτερικές parsed οντότητες που αναφέρονται στο έγγραφο.

Μη επικυρωμένοι επεξεργαστές πρέπει να ελέγχουν μόνο το έγγραφο-οντότητα, συμπεριλαμβανομένου ολόκληρου του εσωτερικού DTD υποσυνόλου, για well-formedness. Καθώς δεν απαιτείται να ελέγξουν το έγγραφο για επικύρωση, πρέπει να επεξεργαστούν όλες τις δηλώσεις που διαβάζουν στο εσωτερικό DTD υποσύνολο και μέσα σε κάθε παράμετρο-οντότητα που διαβάζουν, ως την πρώτη παραπομπή σε παράμετρο-οντότητα που δεν διαβάζουν, με άλλα λόγια, πρέπει να χρησιμοποιούν τις πληροφορίες σε αυτές τις δηλώσεις για να ομαλοποιήσουν τις τιμές των χαρακτηριστικών, να περιλαμβάνουν το κείμενο αντικατάστασης εσωτερικών οντοτήτων, και να παρέχουν default τιμές χαρακτηριστικών. Δεν πρέπει να επεξεργάζονται δηλώσεις οντοτήτων ή δηλώσεις λιστών χαρακτηριστικών που συναντήθηκαν μετά από παραπομπή σε παράμετρο-οντότητα που δεν είναι πραγματική, αφού η οντότητα μπορεί να περιέχει δηλώσεις που υπερισχύουν.

5.2 Χρησιμοποιώντας XML επεξεργαστές.

Η συμπεριφορά του επικυρωμένου επεξεργαστή XML είναι πολύ προβλέψιμη, πρέπει να διαβάζει κάθε κομμάτι από ένα έγγραφο και καταγράφουν παραβιάσεις. Λιγότερα απαιτούνται από ένα όχι επικυρωμένο επεξεργαστή, δεν χρειάζεται

να διαβάζει κάθε κομμάτι του εγγράφου εκτός του εγγράφου-οντότητα. Αυτό έχει δυο αποτελέσματα που μπορεί να είναι σημαντικά για τους χρήστες XML επεξεργαστών:

- Συγκεκριμένα errors καλής μορφοποίησης, ιδιαίτερα αυτά που χρειάζονται διάβασμα εξωτερικής οντότητας, που μπορεί να μην ανακαλυφθούν από μη επικυρωμένους επεξεργαστές.
- Οι πληροφορίες που περνούν από τον επεξεργαστή στην εφαρμογή μπορεί να αλλάξουν, ανάλογα με το κατά πόσο ο επεξεργαστής διαβάζει τις παραμέτρους και εξωτερικές οντότητες. Για παράδειγμα, ένας μη επικυρωμένος επεξεργαστής μπορεί να μην ομαλοποιεί τις τιμές των χαρακτηριστικών, να μην περιλαμβάνει το κείμενο αντικατάστασης εσωτερικών μεταβλητών, ή να παρέχει default τιμές χαρακτηριστικών, όταν η πράξη αυτή εξαρτάται από το αν έχουν διαβαστεί οι δηλώσεις στις εξωτερικές και παραμέτρους- οντότητες.

Για να αυξηθεί η αξιοπιστία στο interoperating μεταξύ διαφορετικών XML επεξεργαστών, λειτουργιών που χρησιμοποιούν μη-επικυρωμένους επεξεργαστές δεν πρέπει να βασίζονται σε συμπεριφορές που δεν ζητούνται από τέτοιους επεξεργαστές. Εφαρμογές που απαιτούν ευκολίες όπως τη χρήση default τιμών χαρακτηριστικών ή εσωτερικές οντότητες που δηλώθηκαν σε εσωτερικές οντότητες πρέπει να χρησιμοποιούν επικυρωμένους XML επεξεργαστές.

6. Notation. (παράσταση)

Η επίσημη γραμματική της XML δίνεται σε αυτό τον προσδιορισμό χρησιμοποιώντας μια απλή Extended Backus-Naur μορφή παράστασης. Κάθε κανόνας στην γραμματική ορίζει ένα σύμβολο στην μορφή

symbol ::= expression

Σύμβολα γράφονται μέσα σε αρχικό κεφαλαίο γράμμα αν ορίζονται από συνήθεις εκφράσεις ή με ένα αρχικό πεζό γράμμα. Τα αρχικά strings αναφέρονται.

Μέσα από την έκφραση του κανόνα right-hand side, οι παρακάτω εκφράσεις που χρησιμοποιούνται για να ταιριάζουν strings ενός ή περισσότερων χαρακτήρων:

#xN

όπου το N είναι hexadecimal integer, η έκφραση ταιριάζει το χαρακτήρα σε ISO/IEC 10646 του οποίου η τιμή κανονικού (UCS-4) κώδικα, όταν διακόπτεται σαν unsigned δυαδικό νούμερο, έχει τιμή καθορισμένη. Το νούμερο που τείνει στο 0 στην μορφή #xN είναι ασήμαντος, το νούμερο που τείνει στο 0 στην τιμή του corresponding κώδικα κυβερνάται από τους χαρακτήρες κωδικοποίησης που χρησιμοποιούνται και δεν είναι σημαντικοί για την XML.

[a-zA-Z], [#xN-#xN]: ταιριάζει κάθε χαρακτήρα με μια τιμή με την σειρά που υποδεικνύεται.

[â-z], [^#xN-#xN]: ταιριάζει κάθε χαρακτήρα με μια τιμή εκτός της σειρά που υποδεικνύεται.

[âbc], [#xN#xN#xN]: ταιριάζει κάθε χαρακτήρα με μια τιμή όχι μεταξύ των δοσμένων χαρακτήρων.

"string": ταιριάζει ένα string με αυτό που βρίσκεται μέσα στο " "

'string': ταιριάζει ένα string με αυτό που βρίσκεται μέσα στο ' '

Αυτά τα σύμβολα μπορεί να συνδυαστούν για να ταιριάζουν με πιο περίπλοκα πρότυπα όπως παρακάτω όπου A και B αντιπροσωπεύουν απλές εκφράσεις :

(expression): το expression συμπεριφέρεται σαν ενότητα και μπορεί να συνδυαστεί όπως περιγράφεται σε αυτή τη λίστα.

A? : Ταιριάζει το A ή τίποτα, προαιρετικά το A

A B : Ταιριάζει το A ακολουθούμενο από B

A | B : Ταιριάζει το A ή το B αλλά όχι και τα δυο

A - B : Ταιριάζει οποίο string ταιριάζει με το A αλλά όχι με το B

A+ : Ταιριάζει ένα ή περισσότερα γεγονότα του A

A* : Ταιριάζει 0 ή περισσότερα γεγονότα του A

Άλλες παραστάσεις χρησιμοποιούνται στην παραγωγή είναι:

/* ... */ : σχόλιο

[wfc : ...] : well-formedness ανάγκη, αυτό αναγνωρίζουν από το όνομα μια ανάγκη σε ένα well-formed έγγραφο σχετικό με το προϊόν.

[vc : ...] : ισχύς ανάγκης, αυτό αναγνωρίζει από το όνομα μια ανάγκη σε έγκυρο έγγραφο σχετικό με ένα προϊόν.

ΠαραρτήματαΑ. Κλάσεις χαρακτήρων

Ακολουθώντας τα χαρακτηριστικά που ορίστηκαν στο Unicode στάνταρ, οι χαρακτήρες μπήκαν σε κλάσεις σαν βασικοί χαρακτήρες, ιδεογραφικοί χαρακτήρες, συνδυαστικοί χαρακτήρες, αυτές οι κλάσεις συνδυάζονται με την κλάση των γραμμάτων. Ψηφία και extenders επίσης διακρίνονται.

Characters

[84]	Letter	::=	BaseChar Ideographic [#x0041-#x005A] [#x0061-#x007A] [#x00C0-#x00D6] [#x00D8-#x00F6] [#x00F8-#x00FF] [#x0100-#x0131] [#x0134-#x013E] [#x0141-#x0148] [#x014A-#x017E] [#x0180-#x01C3] [#x01CD-#x01F0] [#x01F4-#x01F5] [#x01FA-#x0217] [#x0250-#x02A8] [#x02BB-#x02C1] #x0386 [#x0388-#x038A] #x038C [#x038E-#x03A1] [#x03A3-#x03CE] [#x03D0-#x03D6] #x03DA #x03DC #x03DE #x03E0 [#x03E2-#x03F3] [#x0401-#x040C] [#x040E-#x044F] [#x0451-#x045C] [#x045E-#x0481] [#x0490-#x04C4] [#x04C7-#x04C8] [#x04CB-#x04CC] [#x04D0-#x04EB] [#x04EE-#x04F5] [#x04F8-#x04F9] [#x0531-#x0556] #x0559 [#x0561-#x0586] [#x05D0-#x05EA] [#x05F0-#x05F2] [#x0621-#x063A] [#x0641-#x064A] [#x0671-#x06B7] [#x06BA-#x06BE] [#x06C0-#x06CE] [#x06D0-#x06D3] #x06D5 [#x06E5-#x06E6] [#x0905-#x0939] #x093D [#x0958-#x0961] [#x0985-#x098C] [#x098F-#x0990] [#x0993-#x09A8] [#x09AA-#x09B0] #x09B2 [#x09B6-#x09B9] [#x09DC-#x09DD] [#x09DF-#x09E1] [#x09F0-#x09F1] [#x0A05-#x0A0A] [#x0A0F-#x0A10] [#x0A13-#x0A28] [#x0A2A-#x0A30] [#x0A32-#x0A33] [#x0A35-#x0A36] [#x0A38-#x0A39] [#x0A59-#x0A5C] #x0A5E [#x0A72-#x0A74] [#x0A85-#x0A8B] #x0A8D [#x0A8F-#x0A91] [#x0A93-#x0AA8] [#x0AAA-#x0AB0] [#x0AB2-#x0AB3] [#x0AB5-#x0AB9] #x0ABD #x0AE0 [#x0B05-#x0B0C] [#x0B0F-#x0B10] [#x0B13-#x0B28] [#x0B2A-#x0B30] [#x0B32-#x0B33] [#x0B36-#x0B39] #x0B3D [#x0B5C-#x0B5D] [#x0B5F-#x0B61] [#x0B85-#x0B8A] [#x0B8E-#x0B90] [#x0B92-#x0B95] [#x0B99-#x0B9A] #x0B9C [#x0B9E-#x0B9F] [#x0BA3-#x0BA4] [#x0BA8-#x0BAA] [#x0BAE-#x0BB5] [#x0BB7-#x0BB9] [#x0C05-#x0C0C] [#x0C0E-#x0C10] [#x0C12-#x0C28] [#x0C2A-#x0C33] [#x0C35-#x0C39] [#x0C60-#x0C61] [#x0C85-#x0C8C] [#x0C8E-#x0C90] [#x0C92-#x0CA8] [#x0CAA-#x0CB3] [#x0CB5-#x0CB9] #x0CDE [#x0CE0-#x0CE1] [#x0D05-#x0D0C] [#x0D0E-#x0D10] [#x0D12-#x0D28] [#x0D2A-#x0D39] [#x0D60-#x0D61] [#x0E01-#x0E2E] #x0E30 [#x0E32-#x0E33] [#x0E40-#x0E45] [#x0E81-#x0E82] #x0E84 [#x0E87-#x0E88] #x0E8A #x0E8D [#x0E94-#x0E97] [#x0E99-#x0E9F] [#x0EA1-#x0EA3] #x0EA5 #x0EA7 [#x0EAA-#x0EAB] [#x0EAD-#x0EAE] #x0EB0 [#x0EB2-#x0EB3] #x0EBD [#x0EC0-#x0EC4] [#x0F40-#x0F47] [#x0F49-#x0F69] [#x10A0-#x10C5] [#x10D0-#x10F6] #x1100 [#x1102-#x1103] [#x1105-#x1107] #x1109 [#x110B-#x110C] [#x110E-#x1112] #x113C #x113E #x1140 #x114C #x114E #x1150 [#x1154-#x1155] #x1159 [#x115F-#x1161] #x1163 #x1165 #x1167 #x1169 [#x116D-#x116E] [#x1172-#x1173] #x1175 #x119E #x11A8 #x11AB [#x11AE-#x11AF] [#x11B7-#x11B8] #x11BA [#x11BC-#x11C2] #x11EB #x11F0 #x11F9 [#x1E00-#x1E9B] [#x1EA0-#x1EF9] [#x1F00-#x1F15] [#x1F18-#x1F1D] [#x1F20-#x1F45] [#x1F48-#x1F4D] [#x1F50-#x1F57] #x1F59 #x1F5B #x1F5D [#x1F5F-#x1F7D] [#x1F80-#x1FB4] [#x1FB6-#x1FBC] #x1FBE [#x1FC2-#x1FC4] [#x1FC6-#x1FCC] [#x1FD0-#x1FD3] [#x1FD6-#x1FDB] [#x1FE0-#x1FEC] [#x1FF2-#x1FF4] [#x1FF6-#x1FFC] #x2126 [#x212A-#x212B] #x212E [#x2180-#x2182] [#x3041-#x3094] [#x30A1-
[85]	BaseChar	::=	[#x0041-#x005A] [#x0061-#x007A] [#x00C0-#x00D6] [#x00D8-#x00F6] [#x00F8-#x00FF] [#x0100-#x0131] [#x0134-#x013E] [#x0141-#x0148] [#x014A-#x017E] [#x0180-#x01C3] [#x01CD-#x01F0] [#x01F4-#x01F5] [#x01FA-#x0217] [#x0250-#x02A8] [#x02BB-#x02C1] #x0386 [#x0388-#x038A] #x038C [#x038E-#x03A1] [#x03A3-#x03CE] [#x03D0-#x03D6] #x03DA #x03DC #x03DE #x03E0 [#x03E2-#x03F3] [#x0401-#x040C] [#x040E-#x044F] [#x0451-#x045C] [#x045E-#x0481] [#x0490-#x04C4] [#x04C7-#x04C8] [#x04CB-#x04CC] [#x04D0-#x04EB] [#x04EE-#x04F5] [#x04F8-#x04F9] [#x0531-#x0556] #x0559 [#x0561-#x0586] [#x05D0-#x05EA] [#x05F0-#x05F2] [#x0621-#x063A] [#x0641-#x064A] [#x0671-#x06B7] [#x06BA-#x06BE] [#x06C0-#x06CE] [#x06D0-#x06D3] #x06D5 [#x06E5-#x06E6] [#x0905-#x0939] #x093D [#x0958-#x0961] [#x0985-#x098C] [#x098F-#x0990] [#x0993-#x09A8] [#x09AA-#x09B0] #x09B2 [#x09B6-#x09B9] [#x09DC-#x09DD] [#x09DF-#x09E1] [#x09F0-#x09F1] [#x0A05-#x0A0A] [#x0A0F-#x0A10] [#x0A13-#x0A28] [#x0A2A-#x0A30] [#x0A32-#x0A33] [#x0A35-#x0A36] [#x0A38-#x0A39] [#x0A59-#x0A5C] #x0A5E [#x0A72-#x0A74] [#x0A85-#x0A8B] #x0A8D [#x0A8F-#x0A91] [#x0A93-#x0AA8] [#x0AAA-#x0AB0] [#x0AB2-#x0AB3] [#x0AB5-#x0AB9] #x0ABD #x0AE0 [#x0B05-#x0B0C] [#x0B0F-#x0B10] [#x0B13-#x0B28] [#x0B2A-#x0B30] [#x0B32-#x0B33] [#x0B36-#x0B39] #x0B3D [#x0B5C-#x0B5D] [#x0B5F-#x0B61] [#x0B85-#x0B8A] [#x0B8E-#x0B90] [#x0B92-#x0B95] [#x0B99-#x0B9A] #x0B9C [#x0B9E-#x0B9F] [#x0BA3-#x0BA4] [#x0BA8-#x0BAA] [#x0BAE-#x0BB5] [#x0BB7-#x0BB9] [#x0C05-#x0C0C] [#x0C0E-#x0C10] [#x0C12-#x0C28] [#x0C2A-#x0C33] [#x0C35-#x0C39] [#x0C60-#x0C61] [#x0C85-#x0C8C] [#x0C8E-#x0C90] [#x0C92-#x0CA8] [#x0CAA-#x0CB3] [#x0CB5-#x0CB9] #x0CDE [#x0CE0-#x0CE1] [#x0D05-#x0D0C] [#x0D0E-#x0D10] [#x0D12-#x0D28] [#x0D2A-#x0D39] [#x0D60-#x0D61] [#x0E01-#x0E2E] #x0E30 [#x0E32-#x0E33] [#x0E40-#x0E45] [#x0E81-#x0E82] #x0E84 [#x0E87-#x0E88] #x0E8A #x0E8D [#x0E94-#x0E97] [#x0E99-#x0E9F] [#x0EA1-#x0EA3] #x0EA5 #x0EA7 [#x0EAA-#x0EAB] [#x0EAD-#x0EAE] #x0EB0 [#x0EB2-#x0EB3] #x0EBD [#x0EC0-#x0EC4] [#x0F40-#x0F47] [#x0F49-#x0F69] [#x10A0-#x10C5] [#x10D0-#x10F6] #x1100 [#x1102-#x1103] [#x1105-#x1107] #x1109 [#x110B-#x110C] [#x110E-#x1112] #x113C #x113E #x1140 #x114C #x114E #x1150 [#x1154-#x1155] #x1159 [#x115F-#x1161] #x1163 #x1165 #x1167 #x1169 [#x116D-#x116E] [#x1172-#x1173] #x1175 #x119E #x11A8 #x11AB [#x11AE-#x11AF] [#x11B7-#x11B8] #x11BA [#x11BC-#x11C2] #x11EB #x11F0 #x11F9 [#x1E00-#x1E9B] [#x1EA0-#x1EF9] [#x1F00-#x1F15] [#x1F18-#x1F1D] [#x1F20-#x1F45] [#x1F48-#x1F4D] [#x1F50-#x1F57] #x1F59 #x1F5B #x1F5D [#x1F5F-#x1F7D] [#x1F80-#x1FB4] [#x1FB6-#x1FBC] #x1FBE [#x1FC2-#x1FC4] [#x1FC6-#x1FCC] [#x1FD0-#x1FD3] [#x1FD6-#x1FDB] [#x1FE0-#x1FEC] [#x1FF2-#x1FF4] [#x1FF6-#x1FFC] #x2126 [#x212A-#x212B] #x212E [#x2180-#x2182] [#x3041-#x3094] [#x30A1-

		#x30FA] [#x3105-#x312C] [#xAC00-#xD7A3]
[86]	Ideographic ::=	[#x4E00-#x9FA5] #x3007 [#x3021-#x3029] [#x0300-#x0345] [#x0360-#x0361] [#x0483-#x0486] [#x0591-#x05A1] [#x05A3-#x05B9] [#x05BB-#x05BD] #x05BF [#x05C1-#x05C2] #x05C4 [#x064B-#x0652] #x0670 [#x06D6-#x06DC] [#x06DD-#x06DF] [#x06E0- #x06E4] [#x06E7-#x06E8] [#x06EA-#x06ED] [#x0901-#x0903] #x093C [#x093E-#x094C] #x094D [#x0951-#x0954] [#x0962-#x0963] [#x0981- #x0983] #x09BC #x09BE #x09BF [#x09C0-#x09C4] [#x09C7-#x09C8] [#x09CB-#x09CD] #x09D7 [#x09E2-#x09E3] #x0A02 #x0A3C #x0A3E #x0A3F [#x0A40-#x0A42] [#x0A47-#x0A48] [#x0A4B-#x0A4D] [#x0A70-#x0A71] [#x0A81-#x0A83] #x0ABC [#x0ABE-#x0AC5] [#x0AC7- #x0AC9] [#x0ACB-#x0ACD] [#x0B01-#x0B03] #x0B3C [#x0B3E-#x0B43] [87] CombiningChar ::= [#x0B47-#x0B48] [#x0B4B-#x0B4D] [#x0B56-#x0B57] [#x0B82-#x0B83] [#x0BBE-#x0BC2] [#x0BC6-#x0BC8] [#x0BCA-#x0BCD] #x0BD7 [#x0C01- #x0C03] [#x0C3E-#x0C44] [#x0C46-#x0C48] [#x0C4A-#x0C4D] [#x0C55- #x0C56] [#x0C82-#x0C83] [#x0CBE-#x0CC4] [#x0CC6-#x0CC8] [#x0CCA- #x0CCD] [#x0CD5-#x0CD6] [#x0D02-#x0D03] [#x0D3E-#x0D43] [#x0D46-#x0D48] [#x0D4A-#x0D4D] #x0D57 #x0E31 [#x0E34-#x0E3A] [#x0E47-#x0E4E] #x0EB1 [#x0EB4-#x0EB9] [#x0EBB-#x0EBC] [#x0EC8- #x0ECD] [#x0F18-#x0F19] #x0F35 #x0F37 #x0F39 #x0F3E #x0F3F [#x0F71-#x0F84] [#x0F86-#x0F8B] [#x0F90-#x0F95] #x0F97 [#x0F99- #x0FAD] [#x0FB1-#x0FB7] #x0FB9 [#x20D0-#x20DC] #x20E1 [#x302A- #x302F] #x3099 #x309A [#x0030-#x0039] [#x0660-#x0669] [#x06F0-#x06F9] [#x0966-#x096F] [88] Digit ::= [#x09E6-#x09EF] [#x0A66-#x0A6F] [#x0AE6-#x0AEF] [#x0B66-#x0B6F] [#x0BE7-#x0BEF] [#x0C66-#x0C6F] [#x0CE6-#x0CEF] [#x0D66-#x0D6F] [#x0E50-#x0E59] [#x0ED0-#x0ED9] [#x0F20-#x0F29] [89] Extender ::= #x00B7 #x02D0 #x02D1 #x0387 #x0640 #x0E46 #x0EC6 #x3005 [#x3031-#x3035] [#x309D-#x309E] [#x30FC-#x30FE]

Οι κλάσεις χαρακτήρων που ορίζονται εδώ μπορεί να παραχθούν από του βάση δεδομένων χαρακτήρων Unicode όπως παρακάτω.

- Ονόματα αρχικού χαρακτήρα πρέπει να είναι μιας από τις κατηγορίες L1, Lu, Lo, Lt, N1.
- Ονόματα χαρακτήρων άλλων από όνομα-αρχικού χαρακτήρα πρέπει να έχει μια από τις παρακάτω κατηγορίες Mc, Me, Mn, Lm, ή Nd.
- Χαρακτήρες σε συμβατή περιοχή (πχ ...) δεν επιτρέπονται στα XML ονόματα.
- Χαρακτήρες που έχουν μια γραμματοσειρά ή συμβατή ανάλυση (πχ...) δεν επιτρέπονται.
- Οι παρακάτω χαρακτήρες χρησιμοποιούνται σαν ονόματα αρχικού χαρακτήρα παρά σαν ονόματα χαρακτήρα, γιατί οι ιδιότητες τους κατηγοριοποιούνται σαν Alphabetic: [#x02BB-#x02C1], #x0559, #x06E5, #x06E6.
- Οι χαρακτήρες #x20DD-#x20E0 αποκλείονται ().
- Ο χαρακτήρας #x00B7 κατηγοριοποιείται σαν εκτεταμένος, γιατί η λίστα ιδιοτήτων έτσι τον ορίζει.
- Ο χαρακτήρας #x0387 προστίθεται σαν ένα όνομα χαρακτήρα, γιατί το #x00B7 είναι κανονικά ισοδύναμο.
- Οι χαρακτήρες ':' και '_' επιτρέπονται σαν ονόματα αρχικών χαρακτήρων.
- Οι χαρακτήρες '-' και '.' Επιτρέπονται σαν ονόματα χαρακτήρων.

B XML και SGML (non-normative).

Η XML σχεδιάστηκε για να είναι subset της SGML, έτσι κάθε έγκυρο έγγραφο XML πρέπει επίσης να είναι ένα conformant SGML έγγραφο.

Γ Διαστολή οντότητας και παραπομπές χαρακτήρων. (Non-normative).

Αυτό το παράρτημα περιέχει κάποια παραδείγματα που επεξηγούν την αλληλουχία των παραπομπών-οντοτήτων και παραπομπών-χαρακτήρων αναγνώριση και επέκταση όπως ορίστηκαν στο "4.XML Processor Treatment of Entities and References".

Αν το DTD περιέχει την δήλωση

<!ENTITY example "<p>An ampersand (&#38;) may be escaped numerically (&#38;#38;) or with a general entity (&);.</p>" >

τότε ο XML επεξεργαστής θα αναγνωρίζει τις παραπομπές-χαρακτήρα όταν parses την δήλωση της οντότητας και τις αναλύσει πριν αποθηκευτεί το ακόλουθο string σαν τιμή της οντότητας "example":

<p>An ampersand (&) may be escaped numerically (&#38;) or with a general entity (&);.</p>

μια παραπομπή μέσα στο έγγραφο στο "&example;" θα προκαλέσει το έγγραφο να reparsed, ταυτόχρονα οι ετικέτες αρχής και τέλους του στοιχείου "p" θα αναγνωρίζονται και επεκτείνονται, έχοντας σαν αποτέλεσμα στο στοιχείο "p" με το ακόλουθο περιεχόμενο (όλα τα δεδομένα, όχι οριοθέτες ή markup):

An ampersand (&) may be escaped numerically (&) or with a general entity (&);.

Ένα πιο περίπλοκο παράδειγμα θα εξηγήσει τους κανόνες και τα πλήρη αποτελέσματα τους. Στο παρακάτω παράδειγμα, η γραμμή αριθμών είναι μόνο για αναφορά.

```
1 <?xml version='1.0'?>
2 <!DOCTYPE test [
3 <!ELEMENT test (#PCDATA) >
4 <!ENTITY % xx '&#37;zz;*>
5 <!ENTITY % zz '&#60;!ENTITY tricky "error-prone" *>*>
6 %xx;
7 ]>
8 <test>This sample shows a &tricky; method.</test>
```

Αυτό παράγει τα παρακάτω:

- Στη γραμμή 4, η παραπομπή στο χαρακτήρα 37 επεκτείνεται αμέσως, και η παράμετρος οντότητα "xx" αποθηκεύεται στον πίνακα συμβολών με την τιμή "%zz;". Αφού το κείμενο αντικατάστασης δεν επανεξετάζεται, η παραπομπή σε παραμέτρους οντότητες "zz" δεν αναγνωρίζεται. (και θα ήταν λάθος αφού το "zz" δεν έχει δηλωθεί ακόμα.)
- Στη γραμμή 5, η παραπομπή χαρακτήρα "<" επεκτείνεται αμέσως, και η παράμετρος οντότητα "zz" αποθηκεύεται με το κείμενο αντικατάστασης "<! ENTITY tricky "error-prone" *>*>" που είναι μια well-formed δήλωση οντότητας.
- Στη γραμμή 6, η παραπομπή στο "xx" αναγνωρίζεται, και το κείμενο αντικατάστασης του "xx" (ονόματι "%zz;") parsed. Η παραπομπή στο "zz" αναγνωρίζεται με την σειρά της και το κείμενο αντικατάστασης της "<! ENTITY tricky "error-prone" *>*>" parsed. Η γενική οντότητα «tricky» έχει τώρα δηλωθεί, με το κείμενο αντικατάστασης "error-prone".
- Στη γραμμή 8, η παραπομπή στη γενική οντότητα "tricky" αναγνωρίζεται και, επεκτείνεται έτσι το πλήρες περιεχόμενο του στοιχείου "test" είναι το αυτοπεριγραφόμενο string *This sample shows a error-prone method.*

Δ Deterministic Content Models (Non-Normative)

Για συμβατότητα, απαιτείται τα μοντέλα περιεχομένου στην δήλωση τύπων στοιχείου να είναι deterministic.

Η SGML απαιτεί deterministic μοντέλα περιεχομένου (που καλούνται "unambiguous"), οι XML επεξεργαστές που χτίζουν χρησιμοποιώντας SGML συστήματα μπορεί να μαρκάρουν non-deterministic μοντέλα περιεχομένου σαν λάθος. Για παράδειγμα, το μοντέλο περιεχομένου ((b, c) | (b, d)) είναι non-deterministic, γιατί δίνοντας ένα αρχικό b το parser δεν μπορεί να ξέρει ποιο b στο μοντέλο ταιριάζει χωρίς να βλέπει παραπάνω για να δει ποιο στοιχείο ακολουθεί το b. σε αυτή την περίπτωση, οι δυο παραπομπές σε b μπορεί να καταρρεύσουν σε μια παραπομπή κάνοντας το μοντέλο να διαβάσει (b, (c | d)). Ένα αρχικό b τώρα φαίνεται καθαρά ότι ταιριάζει μόνο με ένα απλό όνομα σε μοντέλο περιεχομένου. Το parser δεν χρειάζεται να κοιτά παραπάνω για να δει τι ακολουθεί, μόνο το c ή d θα γίνουν δεκτά. Πιο επίσημα : ένα πεπερασμένο αυτόματο στάδιο μπορεί να κατασκευαστεί από το μοντέλο περιεχομένου χρησιμοποιώντας στάνταρ αλγόριθμους, πχ αλγόριθμος 3.5 στην ενότητα 3.9 του Aho, Sethi, and Ullman [Aho/Ullman]. Σε πολλούς τέτοιους αλγόριθμους, ένα ακόλουθο σεντ κατασκευάζεται για κάθε θέση στην συνηθισμένη έκφραση (πχ), αν κάποια θέση έχει ακόλουθο σεντ στο οποίο περισσότερες από μια ακόλουθες θέσεις έχουν πάρει ετικέτες με το ίδιο στοιχείο τύπου ονόματος, τότε το μοντέλο περιεχομένου είναι λάθος και μπορεί να καταγράφει σαν τέτοιο. Αλγόριθμοι υπάρχουν και επιτρέπουν πολλά αλλά όχι όλα non-deterministic μοντέλα περιεχομένου να υποβιβαστούν αυτόματα σε deterministic μοντέλα.

E Autodetection of Character Encodings (Non-Normative).

Η κωδικοποίηση δήλωσης function στην XML σαν εσωτερική ετικέτα σε κάθε οντότητα, υποδεικνύει ποιος χαρακτήρας κωδικοποίησης χρησιμοποιείται. Πρώτου ένας XML επεξεργαστής μπορεί να διαβάσει την εσωτερική ετικέτα, προφανώς θα πρέπει να γνωρίζει ποιος κωδικοποιημένος χαρακτήρας χρησιμοποιείται—κάτι που η εσωτερική ετικέτα προσπαθεί να δείξει. Σε γενική περίπτωση, αυτό είναι μια χωρίς ελπίδα κατάσταση. Δεν είναι τελείως χωρίς ελπίδα στην XML γιατί η XML περιορίζει την κατάσταση με δυο τρόπους: κάθε εκτέλεση (εφαρμογή) αναλαμβάνει να υποστηρίξει μόνο ένα καθορισμένο σετ χαρακίων κωδικοποίησης, και XML κωδικοποίηση της δήλωσης είναι περιορισμένη σε θέση και περιεχόμενο προκειμένου να γίνει πραγματοποιήσιμο η autodetect της κωδικοποίησης χαρακτήρα στη χρήση κάθε οντότητας σε κανονικές περιπτώσεις. Επίσης σε πολλές περιπτώσεις μπορεί να διαχωρίζονται, ανάλογα με το αν η XML οντότητα παρουσιάζεται στον επεξεργαστή χωρίς ή με συνοδευτικές (εξωτερικές) πληροφορίες. Θεωρούμε την πρώτη περίπτωση πρώτα.Επειδή κάθε XML οντότητα όχι σε UTF-8 ή UTF-16 format πρέπει να ξεκινά με δήλωση XML κωδικοποίησης, στην οποία ο πρώτος χαρακτήρας πρέπει να είναι '<?xml' κάθε προσαρμοσμένος επεξεργαστής μπορεί να εντοπίζει, κάθε 2 ή 4 οντάδες δεδομένων εισόδου, ποια από τις παρακάτω περιπτώσεις εφαρμόζει, διαβάζοντας αυτή την λίστα, θα βοηθούσε να ξέρουμε ότι στο UCS-4, το 'ξ' είναι το "x0000003C" και το 'F' είναι το «#x0000003F», και το Byte Order Mark που απαιτείται από το UTF-16 data streams είναι "#xFEFF".

- 00 00 00 3C: UCS-4, big-endian machine (1234 order)
- 3C 00 00 00: UCS-4, little-endian machine (4321 order)
- 00 00 3C 00: UCS-4, unusual octet order (2143)
- 00 3C 00 00: UCS-4, unusual octet order (3412)
- FE FF: UTF-16, big-endian
- FF FE: UTF-16, little-endian
- 00 3C 00 3F: UTF-16, big-endian, no Byte Order Mark (and thus, strictly speaking, in error)
- 3C 00 3F 00: UTF-16, little-endian, no Byte Order Mark (and thus, strictly speaking, in error)
- 3C 3F 78 6D: UTF-8, ISO 646, ASCII, some part of ISO 8859, Shift-JIS, EUC, or any other 7-bit, 8-bit, or mixed-width encoding που εξασφαλίζουν ότι οι χαρακτήρες ASCII έχουν την κανονική τους θέση, πλάτος, και τιμή, η δήλωση πραγματικής κωδικοποίησης πρέπει να διαβαστεί ώστε να προσδιοριστεί ποια από αυτές εφαρμόζει, αλλά όλες αυτές οι κωδικοποιήσεις χρησιμοποιούν bit πρότυπα για ASCII χαρακτήρες, η δήλωση κωδικοποίησης μπορεί να διαβαστεί αξιόπιστα.
- 4C 6F A7 94: EBCDIC (in some flavor; the full encoding declaration must be read to tell which code page is in use)
- other: UTF-8 without an encoding declaration, or else the data stream is corrupt, fragmentary, or enclosed in a wrapper of some kind

αυτό το επίπεδο autodetection είναι αρκετό για να διαβάσει την δήλωση κωδικοποίησης XML και να parse το identifier χαρακτήρα-κωδικοποίησης, το οποίο είναι ακόμα απαραίτητα για να διαχωρίζουμε τα ξεχωριστά μέλη από κάθε οικογένεια κωδικοποίησης.Επειδή τα περιεχόμενα της δήλωσης κωδικοποίησης είναι περιορισμένα σε ASCII χαρακτήρες, ένας επεξεργαστής μπορεί αξιόπιστα να διαβάσει ολόκληροι τη δήλωση κωδικοποίησης αμέσως μόλις προσδιορίσει ποια οικογένεια κωδικοποίησης χρησιμοποιείται. Αφού στην πράξη, όλοι ευρέως χρησιμοποιούν χαρακτήρες κωδικοποίησης που ανήκουν σε μια από τις κατηγορίες που είπαμε παραπάνω, η δήλωση κωδικοποίησης στην XML επιτρέπει λογικά αξιόπιστη in-band ετικετοποίηση της κωδικοποίησης χαρακτήρων, ακόμα και όταν εξωτερικές πηγές πληροφοριών από λειτουργικό σύστημα ή το επίπεδο πρωτοκόλλου-μεταφοράς είναι αναξιόπιστα.Μόλις ο επεξεργαστής προσδιορίσει το χαρακτήρα κωδικοποίησης που χρησιμοποιείται, μπορεί να πράξει ανάλογα, είτε να καλέσει μια ξεχωριστή Input ρουτίνα για κάθε περίπτωση, ή να καλέσει την κατάλληλη συνάρτηση μετατροπής για κάθε χαρακτήρα του input.

Όπως κάθε Self-labeling σύστημα, η XML δήλωση κωδικοποίησης δεν θα λειτουργήσει αν κάποιο software αλλάξει το σετ χαρακτήρων της οντότητας ή κωδικοποιήσει χωρίς να κάνει Update την δήλωση κωδικοποίησης. Οι χρήστες των ρουτινών κωδικοποίησης-χαρακτήρων πρέπει να είναι προσεκτικοί για να εξασφαλίσουν την ορθότητα των εσωτερικών και εξωτερικών πληροφοριών που χρησιμοποιούνται για τις ετικέτες των οντοτήτων.

Η δεύτερη πιθανή περίπτωση τυχαίνει όταν η XML οντότητα συνοδεύεται από πληροφορίες κωδικοποίησης, όπως σε κάποια συστήματα και κάποια δικτυακά πρωτοκόλλα. Όταν πολλαπλές πηγές πληροφοριών είναι διαθέσιμες, η σχετική του προτεραιότητα και η προτιμότερη μέθοδος για το χειρισμό συγκρούσεων πρέπει να προσδιοριστεί σαν μέρος των higher-level πρωτοκόλλα που χρησιμοποιούνται για την παράδοση XML. Κανόνες για την σχετική προτεραιότητα των εσωτερικών ετικετών και η MIME-type ετικέτα σε μια εξωτερική επικεφαλίδα, για παράδειγμα, πρέπει να είναι μέρος

του RFC έγγραφο ορίζοντας τους text/xml και application/xml MIME τύπους. Στα ενδιαφέροντα του interoperability πάντως, οι παρακάτω κανόνες προτείνονται.

- Αν μια XML οντότητα είναι μέσα σε ένα αρχείο, το Byte-Order Mark και η δήλωση κωδικοποίησης PI χρησιμοποιούνται για να ορίσουν το χαρακτήρα κωδικοποίησης. Όλες οι άλλες επαγωγικές μέθοδοι και πηγές πληροφοριών μόνο για error recovery.
- Αν μια XML οντότητα έχει παραδοθεί με MIME τύπο από text/xml, τότε η παράμετρος charset στον τύπο MIME ορίζει τη μέθοδο χαρακτήρα κωδικοποίησης. Όλες οι άλλες επαγωγικές μέθοδοι και πηγές πληροφοριών μόνο για error recovery.
- Αν μια XML οντότητα έχει παραδοθεί με MIME τύπο από application /xml, τότε το Byte-Order Mark και η δήλωση κωδικοποίησης PI χρησιμοποιούνται για να ορίσουν το χαρακτήρα κωδικοποίησης. Όλες οι άλλες επαγωγικές μέθοδοι και πηγές πληροφοριών μόνο για error recovery.

Αυτοί οι κανόνες εφαρμόζονται μόνο στην απουσία protocol-level documentation. Πρακτικά, όταν οι τύποι MIME text/xml και application/xml ορίζονται, οι συστάσεις του σχετικού RFC θα αντικαταστήσουν αυτούς τους κανόνες.