|  |  |  |
| --- | --- | --- |
|  | |  |
|  | | Requirements Document |
|  | |  |
|  | Decoder subsystem  **Version: 1.4**  **Last Revised:** December 30, 2010  **Author: Maksym Sukhovarov** | |

Table of Contents

[Feature Overview 6](#_Toc280863012)

[1 Terminology 6](#_Toc280863013)

[2 Product Requirements 6](#_Toc280863014)

[2.1 HTML decoder 7](#_Toc280863015)

[2.2 HTML attribute decoder 18](#_Toc280863016)

[2.3 URL decoder 18](#_Toc280863017)

[2.4 JavaScript decoder 20](#_Toc280863018)

[2.5 VBScript decoder 22](#_Toc280863019)

[2.6 XML decoder 24](#_Toc280863020)

[2.7 XML attribute decoder 24](#_Toc280863021)

[2.8 LDAP query decoder 24](#_Toc280863022)

[2.9 CSS decoder 25](#_Toc280863023)

[2.10 Base64 decoder 25](#_Toc280863024)

[2.11 GZip decoder 29](#_Toc280863025)

[2.12 Null HTML sanitizer 29](#_Toc280863026)

[3 Functional Area 30](#_Toc280863027)

[Apendix A – Unicode, UTF-16 and UTF-8 encodings 32](#_Toc280863028)

[Appendix A.1 – Unicode characters and UTF-16 encodings 32](#_Toc280863029)

[Appendix A.2 – Unicode characters and UTF-8 encodings 32](#_Toc280863030)

[Appendix A.3 – Unicode characters in hexadecimal notation 34](#_Toc280863031)

Document Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Revision No.** | **Date** | **Description of Change** |
| Maksym Sukhovarov | 1.0 | 11/15/2010 | Initial version |
| Anatoliy Lokshin | 1.1 | 11/26/2010 | Minor fixes. |
| Maksym Sukhovarov | 1.2 | 12/01/2010 | %uUUUU format was added for URL decoder |
| Maksym Sukhovarov | 1.2 | 12/02/2010 | Format was corrected for XML decoder; |
| Anatoliy Lokshin | 1.3 | 12/06/2010 | Changes according to the issue SECFRM-75 (more named entities for the HTML decoder). |
| Anatoliy Lokshin | 1.4 | 12/08/2010 | Fixed info on HTML named entities case sensitivity. |
| Maksym Sukhovarov | 1.5 | 12/23/2010 | Replaced utf-8 encoding to Unicode and UTF-16 Appendix was added. |
| Maksym Sukhovarov | 1.5 | 1/19/2011 | Octal format for JavaScript was added |

Approvers

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Date Approved** |
| **Kyle Quest** | Architect |  |

Reviewers

| **Name** | **Title** | **Date Reviewed** |
| --- | --- | --- |
| **Kyle Quest** | Architect |  |
| **Anatoliy Lokshin** | Development lead | 11/26/2010 |
| **Julia Kuchmai** | QA Representative |  |

References

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Author** | **Location** |
| Base64 | wikipedia | <http://en.wikipedia.org/wiki/Base64> |
| ­Character encodings in HTML | wikipedia | <http://en.wikipedia.org/wiki/Character_encodings_in_HTML> |
| Character encodings in XML and in HTML | wikipedia | <http://en.wikipedia.org/wiki/Character_encodings_in_HTML> |
| Character entity references in HTML 4 | W3.org | <http://www.w3.org/TR/html4/sgml/entities.html> |
| Comparison of Unicode encoding | wikipedia | <http://en.wikipedia.org/wiki/Comparison_of_Unicode_encodings> |
| JavaScript Special Characters |  | <http://www.w3schools.com/js/js_special_characters.asp> |
| Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters | Network Working Group | <http://www.rfc-editor.org/rfc/rfc4515.txt> |
| Lightweight Directory Access Protocol (v3):  UTF-8 String Representation of Distinguished Names | RCF | <http://tools.ietf.org/html/rfc2253> |
| List of XML and HTML character entity references | wikipedia | <http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references> |
| |  | | --- | | MetraTech Security Framework Specification | | Kyle Quest | <http://seceng.metratech.com/gf/download/docmanfileversion/20/78/MtSecurityFrameworkSpec.doc> |
| Uniform Resource Identifier (URI): Generic Syntax | rfc3986 | <http://tools.ietf.org/html/rfc3986> |
| URL-encoding | wikipedia | <http://en.wikipedia.org/wiki/Percent-encoding> |
| URL-encoding |  | <http://www.javascripter.net/faq/escape.htm> |
| Using character escapes in markup and CSS | W3C | <http://www.w3.org/International/questions/qa-escapes#cssescapes> |
| UTF-8 encoding table and Unicode characters |  | <http://www.utf8-chartable.de/> |
| Issue SECFRM-75 Decoder test application doesn't decodes some HTML entity characters | JIRA | <https://jira.metratech.com/browse/SECFRM-75> |

Online tools

| Name | Reference |
| --- | --- |
| Character set converter – online tool (Unicode, utf-8, utf-16, utf-32 …) | <http://kanjidict.stc.cx/recode.php> |
| Unicode Code Converter v7.03 (Web-converter forked only in Firefox browser) | <http://rishida.net/tools/conversion/> |
| UniView (show character codes) | http://rishida.net/scripts/uniview/uniview.php |

# Feature Overview

The Decoder subsystem is responsible for providing the decoding some sequences.

# Terminology

Requirements that are labeled with an R (such as R-400) are Mandatory (or Required) for the targeted release.

Requirements that are labeled with an O (such as O-400) are Optional for the targeted release.

It is left to Engineering’s discretion on whether these can be met for the targeted release but it is imperative that Engineering take these into consideration when making any design or infrastructure decisions. This holds good for any requirement that might be listed under “Future Requirements” section as well.

**(R-1)** It is a Mandatory requirement that the infrastructure developed shall support the ‘feasibility’ of all feature requirements listed in the document – though the delivery of the features themselves could be scheduled for a later time.

# Product Requirements

The Decoder subsystem must contain following categories:

* HTML decoder;
* HTML attribute decoder;
* URL decoder;
* JavaScript decoder;
* VB script decoder;
* XML decoder;
* XML attribute decoder;
* LDAP query decoder;
* CSS decoder;
* Base64 group decoder;
* GZip decoder;

## HTML decoder

HTML decoder must support three formats of HTML decoding ([see ref](#CharacterEncodingsInHTML).):

1. Character entity format **&name[;]**[[1]](#footnote-1), where name is generally case-sensitive alphanumeric string, exceptions are &quot; &amp; &lt; &gt; &copy; &reg; which can be either in lower or in upper case;
2. Hexadecimal format, **&#x[0]\*[[2]](#footnote-2)HHHHHH[;]**, where HHHHHH is a max 6 of hexadecimal digits;
3. Decimal format, **&#[0]\*DDDDDDDD[;]** , where DDDDDDDD is a max 8 of decimal digits;

Character entity references in HTML see below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Character** | **Entity** | **Hex code** | **Decimal code** | **Standard** |
| quot | " | &quot[;] | &#x[0]\*22[;] | &#[0]\*34[;] | HTML 2.0 |
| amp | & | &amp[;] | &#x[0]\*26[;] | &#[0]\*38[;] | HTML 2.0 |
| apos | ' | &apos[;] | &#x[0]\*27[;] | &#[0]\*39[;] | XHTML 1.0 |
| lpar | ( | &lpar[;] | &#x[0]\*28[;] | &#[0]\*40[;] | HTML 2.0 |
| rpar | ) | &rpar[;] | &#x[0]\*29[;] | &#[0]\*41[;] | HTML 2.0 |
| lt | < | &lt[;] | &#x[0]\*3C[;] | &#[0]\*60[;] | HTML 2.0 |
| gt | > | &gt[;] | &#x[0]\*3E[;] | &#[0]\*62[;] | HTML 2.0 |
| nbsp |  | &nbsp[;] | &#x[0]\*A0[;] | &#[0]\*160[;] | HTML 3.2 |
| iexcl | ¡ | &iexcl[;] | &#x[0]\*A1[;] | &#[0]\*161[;] | HTML 3.2 |
| cent | ¢ | &cent[;] | &#x[0]\*A2[;] | &#[0]\*162[;] | HTML 3.2 |
| pound | £ | &pound[;] | &#x[0]\*A3[;] | &#[0]\*163[;] | HTML 3.2 |
| curren | ¤ | &curren[;] | &#x[0]\*A4[;] | &#[0]\*164[;] | HTML 3.2 |
| yen | ¥ | &yen[;] | &#x[0]\*A5[;] | &#[0]\*165[;] | HTML 3.2 |
| brvbar | ¦ | &brvbar[;] | &#x[0]\*A6[;] | &#[0]\*166[;] | HTML 3.2 |
| sect | § | &sect[;] | &#x[0]\*A7[;] | &#[0]\*167[;] | HTML 3.2 |
| uml | ¨ | &uml[;] | &#x[0]\*A8[;] | &#[0]\*168[;] | HTML 3.2 |
| copy | © | &copy[;] | &#x[0]\*A9[;] | &#[0]\*169[;] | HTML 3.2 |
| ordf | ª | &ordf[;] | &#x[0]\*AA[;] | &#[0]\*170[;] | HTML 3.2 |
| laquo | « | &laquo[;] | &#x[0]\*AB[;] | &#[0]\*171[;] | HTML 3.2 |
| not | ¬ | &not[;] | &#x[0]\*AC[;] | &#[0]\*172[;] | HTML 3.2 |
| shy |  | &shy[;] | &#x[0]\*AD[;] | &#[0]\*173[;] | HTML 3.2 |
| reg | ® | &reg[;] | &#x[0]\*AE[;] | &#[0]\*174[;] | HTML 3.2 |
| macr | ¯ | &macr[;] | &#x[0]\*AF[;] | &#[0]\*175[;] | HTML 3.2 |
| deg | ° | &deg[;] | &#x[0]\*B0[;] | &#[0]\*176[;] | HTML 3.2 |
| plusmn | ± | &plusmn[;] | &#x[0]\*B1[;] | &#[0]\*177[;] | HTML 3.2 |
| sup2 | ² | &sup2[;] | &#x[0]\*B2[;] | &#[0]\*178[;] | HTML 3.2 |
| sup3 | ³ | &sup3[;] | &#x[0]\*B3[;] | &#[0]\*179[;] | HTML 3.2 |
| acute | ´ | &acute[;] | &#x[0]\*B4[;] | &#[0]\*180[;] | HTML 3.2 |
| micro | µ | &micro[;] | &#x[0]\*B5[;] | &#[0]\*181[;] | HTML 3.2 |
| para | ¶ | &para[;] | &#x[0]\*B6[;] | &#[0]\*182[;] | HTML 3.2 |
| middot | · | &middot[;] | &#x[0]\*B7[;] | &#[0]\*183[;] | HTML 3.2 |
| cedil | ¸ | &cedil[;] | &#x[0]\*B8[;] | &#[0]\*184[;] | HTML 3.2 |
| sup1 | ¹ | &sup1[;] | &#x[0]\*B9[;] | &#[0]\*185[;] | HTML 3.2 |
| ordm | º | &ordm[;] | &#x[0]\*BA[;] | &#[0]\*186[;] | HTML 3.2 |
| raquo | » | &raquo[;] | &#x[0]\*BB[;] | &#[0]\*187[;] | HTML 3.2 |
| frac14 | ¼ | &frac14[;] | &#x[0]\*BC[;] | &#[0]\*188[;] | HTML 3.2 |
| frac12 | ½ | &frac12[;] | &#x[0]\*BD[;] | &#[0]\*189[;] | HTML 3.2 |
| frac34 | ¾ | &frac34[;] | &#x[0]\*BE[;] | &#[0]\*190[;] | HTML 3.2 |
| iquest | ¿ | &iquest[;] | &#x[0]\*BF[;] | &#[0]\*191[;] | HTML 3.2 |
| Agrave | À | &Agrave[;] | &#x[0]\*C0[;] | &#[0]\*192[;] | HTML 2.0 |
| Aacute | Á | &Aacute[;] | &#x[0]\*C1[;] | &#[0]\*193[;] | HTML 2.0 |
| Acirc | Â | &Acirc[;] | &#x[0]\*C2[;] | &#[0]\*194[;] | HTML 2.0 |
| Atilde | Ã | &Atilde[;] | &#x[0]\*C3[;] | &#[0]\*195[;] | HTML 2.0 |
| Auml | Ä | &Auml[;] | &#x[0]\*C4[;] | &#[0]\*196[;] | HTML 2.0 |
| Aring | Å | &Aring[;] | &#x[0]\*C5[;] | &#[0]\*197[;] | HTML 2.0 |
| AElig | Æ | &AElig[;] | &#x[0]\*C6[;] | &#[0]\*198[;] | HTML 2.0 |
| Ccedil | Ç | &Ccedil[;] | &#x[0]\*C7[;] | &#[0]\*199[;] | HTML 2.0 |
| Egrave | È | &Egrave[;] | &#x[0]\*C8[;] | &#[0]\*200[;] | HTML 2.0 |
| Eacute | É | &Eacute[;] | &#x[0]\*C9[;] | &#[0]\*201[;] | HTML 2.0 |
| Ecirc | Ê | &Ecirc[;] | &#x[0]\*CA[;] | &#[0]\*202[;] | HTML 2.0 |
| Euml | Ë | &Euml[;] | &#x[0]\*CB[;] | &#[0]\*203[;] | HTML 2.0 |
| Igrave | Ì | &Igrave[;] | &#x[0]\*CC[;] | &#[0]\*204[;] | HTML 2.0 |
| Iacute | Í | &Iacute[;] | &#x[0]\*CD[;] | &#[0]\*205[;] | HTML 2.0 |
| Icirc | Î | &Icirc[;] | &#x[0]\*CE[;] | &#[0]\*206[;] | HTML 2.0 |
| Iuml | Ï | &Iuml[;] | &#x[0]\*CF[;] | &#[0]\*207[;] | HTML 2.0 |
| ETH | Ð | &ETH[;] | &#x[0]\*D0[;] | &#[0]\*208[;] | HTML 2.0 |
| Ntilde | Ñ | &Ntilde[;] | &#x[0]\*D1[;] | &#[0]\*209[;] | HTML 2.0 |
| Ograve | Ò | &Ograve[;] | &#x[0]\*D2[;] | &#[0]\*210[;] | HTML 2.0 |
| Oacute | Ó | &Oacute[;] | &#x[0]\*D3[;] | &#[0]\*211[;] | HTML 2.0 |
| Ocirc | Ô | &Ocirc[;] | &#x[0]\*D4[;] | &#[0]\*212[;] | HTML 2.0 |
| Otilde | Õ | &Otilde[;] | &#x[0]\*D5[;] | &#[0]\*213[;] | HTML 2.0 |
| Ouml | Ö | &Ouml[;] | &#x[0]\*D6[;] | &#[0]\*214[;] | HTML 2.0 |
| times | × | &times[;] | &#x[0]\*D7[;] | &#[0]\*215[;] | HTML 3.2 |
| Oslash | Ø | &Oslash[;] | &#x[0]\*D8[;] | &#[0]\*216[;] | HTML 2.0 |
| Ugrave | Ù | &Ugrave[;] | &#x[0]\*D9[;] | &#[0]\*217[;] | HTML 2.0 |
| Uacute | Ú | &Uacute[;] | &#x[0]\*DA[;] | &#[0]\*218[;] | HTML 2.0 |
| Ucirc | Û | &Ucirc[;] | &#x[0]\*DB[;] | &#[0]\*219[;] | HTML 2.0 |
| Uuml | Ü | &Uuml[;] | &#x[0]\*DC[;] | &#[0]\*220[;] | HTML 2.0 |
| Yacute | Ý | &Yacute[;] | &#x[0]\*DD[;] | &#[0]\*221[;] | HTML 2.0 |
| THORN | Þ | &THORN[;] | &#x[0]\*DE[;] | &#[0]\*222[;] | HTML 2.0 |
| szlig | ß | &szlig[;] | &#x[0]\*DF[;] | &#[0]\*223[;] | HTML 2.0 |
| agrave | à | &agrave[;] | &#x[0]\*E0[;] | &#[0]\*224[;] | HTML 2.0 |
| aacute | á | &aacute[;] | &#x[0]\*E1[;] | &#[0]\*225[;] | HTML 2.0 |
| acirc | â | &acirc[;] | &#x[0]\*E2[;] | &#[0]\*226[;] | HTML 2.0 |
| atilde | ã | &atilde[;] | &#x[0]\*E3[;] | &#[0]\*227[;] | HTML 2.0 |
| auml | ä | &auml[;] | &#x[0]\*E4[;] | &#[0]\*228[;] | HTML 2.0 |
| aring | å | &aring[;] | &#x[0]\*E5[;] | &#[0]\*229[;] | HTML 2.0 |
| aelig | æ | &aelig[;] | &#x[0]\*E6[;] | &#[0]\*230[;] | HTML 2.0 |
| ccedil | ç | &ccedil[;] | &#x[0]\*E7[;] | &#[0]\*231[;] | HTML 2.0 |
| egrave | è | &egrave[;] | &#x[0]\*E8[;] | &#[0]\*232[;] | HTML 2.0 |
| eacute | é | &eacute[;] | &#x[0]\*E9[;] | &#[0]\*233[;] | HTML 2.0 |
| ecirc | ê | &ecirc[;] | &#x[0]\*EA[;] | &#[0]\*234[;] | HTML 2.0 |
| euml | ë | &euml[;] | &#x[0]\*EB[;] | &#[0]\*235[;] | HTML 2.0 |
| igrave | ì | &igrave[;] | &#x[0]\*EC[;] | &#[0]\*236[;] | HTML 2.0 |
| iacute | í | &iacute[;] | &#x[0]\*ED[;] | &#[0]\*237[;] | HTML 2.0 |
| icirc | î | &icirc[;] | &#x[0]\*EE[;] | &#[0]\*238[;] | HTML 2.0 |
| iuml | ï | &iuml[;] | &#x[0]\*EF[;] | &#[0]\*239[;] | HTML 2.0 |
| eth | ð | &eth[;] | &#x[0]\*F0[;] | &#[0]\*240[;] | HTML 2.0 |
| ntilde | ñ | &ntilde[;] | &#x[0]\*F1[;] | &#[0]\*241[;] | HTML 2.0 |
| ograve | ò | &ograve[;] | &#x[0]\*F2[;] | &#[0]\*242[;] | HTML 2.0 |
| oacute | ó | &oacute[;] | &#x[0]\*F3[;] | &#[0]\*243[;] | HTML 2.0 |
| ocirc | ô | &ocirc[;] | &#x[0]\*F4[;] | &#[0]\*244[;] | HTML 2.0 |
| otilde | õ | &otilde[;] | &#x[0]\*F5[;] | &#[0]\*245[;] | HTML 2.0 |
| ouml | ö | &ouml[;] | &#x[0]\*F6[;] | &#[0]\*246[;] | HTML 2.0 |
| divide | ÷ | &divide[;] | &#x[0]\*F7[;] | &#[0]\*247[;] | HTML 3.2 |
| oslash | ø | &oslash[;] | &#x[0]\*F8[;] | &#[0]\*248[;] | HTML 2.0 |
| ugrave | ù | &ugrave[;] | &#x[0]\*F9[;] | &#[0]\*249[;] | HTML 2.0 |
| uacute | ú | &uacute[;] | &#x[0]\*FA[;] | &#[0]\*250[;] | HTML 2.0 |
| ucirc | û | &ucirc[;] | &#x[0]\*FB[;] | &#[0]\*251[;] | HTML 2.0 |
| uuml | ü | &uuml[;] | &#x[0]\*FC[;] | &#[0]\*252[;] | HTML 2.0 |
| yacute | ý | &yacute[;] | &#x[0]\*FD[;] | &#[0]\*253[;] | HTML 2.0 |
| thorn | þ | &thorn[;] | &#x[0]\*FE[;] | &#[0]\*254[;] | HTML 2.0 |
| yuml | ÿ | &yuml[;] | &#x[0]\*FF[;] | &#[0]\*255[;] | HTML 2.0 |
| OElig | Œ | &OElig[;] | &#x[0]\*152[;] | &#[0]\*338[;] | HTML 4.0 |
| oelig | œ | &oelig[;] | &#x[0]\*153[;] | &#[0]\*339[;] | HTML 4.0 |
| Scaron | Š | &Scaron[;] | &#x[0]\*160[;] | &#[0]\*352[;] | HTML 4.0 |
| scaron | š | &scaron[;] | &#x[0]\*161[;] | &#[0]\*353[;] | HTML 4.0 |
| Yuml | Ÿ | &Yuml[;] | &#x[0]\*178[;] | &#[0]\*376[;] | HTML 4.0 |
| fnof | ƒ | &fnof[;] | &#x[0]\*192[;] | &#[0]\*402[;] | HTML 4.0 |
| circ | ˆ | &circ[;] | &#x[0]\*2C6[;] | &#[0]\*710[;] | HTML 4.0 |
| tilde | ˜ | &tilde[;] | &#x[0]\*2DC[;] | &#[0]\*732[;] | HTML 4.0 |
| Alpha | Α | &Alpha[;] | &#x[0]\*391[;] | &#[0]\*913[;] | HTML 4.0 |
| Beta | Β | &Beta[;] | &#x[0]\*392[;] | &#[0]\*914[;] | HTML 4.0 |
| Gamma | Γ | &Gamma[;] | &#x[0]\*393[;] | &#[0]\*915[;] | HTML 4.0 |
| Delta | Δ | &Delta[;] | &#x[0]\*394[;] | &#[0]\*916[;] | HTML 4.0 |
| Epsilon | Ε | &Epsilon[;] | &#x[0]\*395[;] | &#[0]\*917[;] | HTML 4.0 |
| Zeta | Ζ | &Zeta[;] | &#x[0]\*396[;] | &#[0]\*918[;] | HTML 4.0 |
| Eta | Η | &Eta[;] | &#x[0]\*397[;] | &#[0]\*919[;] | HTML 4.0 |
| Theta | Θ | &Theta[;] | &#x[0]\*398[;] | &#[0]\*920[;] | HTML 4.0 |
| Iota | Ι | &Iota[;] | &#x[0]\*399[;] | &#[0]\*921[;] | HTML 4.0 |
| Kappa | Κ | &Kappa[;] | &#x[0]\*39A[;] | &#[0]\*922[;] | HTML 4.0 |
| Lambda | Λ | &Lambda[;] | &#x[0]\*39B[;] | &#[0]\*923[;] | HTML 4.0 |
| Mu | Μ | &Mu[;] | &#x[0]\*39C[;] | &#[0]\*924[;] | HTML 4.0 |
| Nu | Ν | &Nu[;] | &#x[0]\*39D[;] | &#[0]\*925[;] | HTML 4.0 |
| Xi | Ξ | &Xi[;] | &#x[0]\*39E[;] | &#[0]\*926[;] | HTML 4.0 |
| Omicron | Ο | &Omicron[;] | &#x[0]\*39F[;] | &#[0]\*927[;] | HTML 4.0 |
| Pi | Π | &Pi[;] | &#x[0]\*3A0[;] | &#[0]\*928[;] | HTML 4.0 |
| Rho | Ρ | &Rho[;] | &#x[0]\*3A1[;] | &#[0]\*929[;] | HTML 4.0 |
| Sigma | Σ | &Sigma[;] | &#x[0]\*3A3[;] | &#[0]\*931[;] | HTML 4.0 |
| Tau | Τ | &Tau[;] | &#x[0]\*3A4[;] | &#[0]\*932[;] | HTML 4.0 |
| Upsilon | Υ | &Upsilon[;] | &#x[0]\*3A5[;] | &#[0]\*933[;] | HTML 4.0 |
| Phi | Φ | &Phi[;] | &#x[0]\*3A6[;] | &#[0]\*934[;] | HTML 4.0 |
| Chi | Χ | &Chi[;] | &#x[0]\*3A7[;] | &#[0]\*935[;] | HTML 4.0 |
| Psi | Ψ | &Psi[;] | &#x[0]\*3A8[;] | &#[0]\*936[;] | HTML 4.0 |
| Omega | Ω | &Omega[;] | &#x[0]\*3A9[;] | &#[0]\*937[;] | HTML 4.0 |
| alpha | α | &alpha[;] | &#x[0]\*3B1[;] | &#[0]\*945[;] | HTML 4.0 |
| beta | β | &beta[;] | &#x[0]\*3B2[;] | &#[0]\*946[;] | HTML 4.0 |
| gamma | γ | &gamma[;] | &#x[0]\*3B3[;] | &#[0]\*947[;] | HTML 4.0 |
| delta | δ | &delta[;] | &#x[0]\*3B4[;] | &#[0]\*948[;] | HTML 4.0 |
| epsilon | ε | &epsilon[;] | &#x[0]\*3B5[;] | &#[0]\*949[;] | HTML 4.0 |
| zeta | ζ | &zeta[;] | &#x[0]\*3B6[;] | &#[0]\*950[;] | HTML 4.0 |
| eta | η | &eta[;] | &#x[0]\*3B7[;] | &#[0]\*951[;] | HTML 4.0 |
| theta | θ | &theta[;] | &#x[0]\*3B8[;] | &#[0]\*952[;] | HTML 4.0 |
| iota | ι | &iota[;] | &#x[0]\*3B9[;] | &#[0]\*953[;] | HTML 4.0 |
| kappa | κ | &kappa[;] | &#x[0]\*3BA[;] | &#[0]\*954[;] | HTML 4.0 |
| lambda | λ | &lambda[;] | &#x[0]\*3BB[;] | &#[0]\*955[;] | HTML 4.0 |
| mu | μ | &mu[;] | &#x[0]\*3BC[;] | &#[0]\*956[;] | HTML 4.0 |
| nu | ν | &nu[;] | &#x[0]\*3BD[;] | &#[0]\*957[;] | HTML 4.0 |
| xi | ξ | &xi[;] | &#x[0]\*3BE[;] | &#[0]\*958[;] | HTML 4.0 |
| omicron | ο | &omicron[;] | &#x[0]\*3BF[;] | &#[0]\*959[;] | HTML 4.0 |
| pi | π | &pi[;] | &#x[0]\*3C0[;] | &#[0]\*960[;] | HTML 4.0 |
| rho | ρ | &rho[;] | &#x[0]\*3C1[;] | &#[0]\*961[;] | HTML 4.0 |
| sigmaf | ς | &sigmaf[;] | &#x[0]\*3C2[;] | &#[0]\*962[;] | HTML 4.0 |
| sigma | σ | &sigma[;] | &#x[0]\*3C3[;] | &#[0]\*963[;] | HTML 4.0 |
| tau | τ | &tau[;] | &#x[0]\*3C4[;] | &#[0]\*964[;] | HTML 4.0 |
| upsilon | υ | &upsilon[;] | &#x[0]\*3C5[;] | &#[0]\*965[;] | HTML 4.0 |
| phi | φ | &phi[;] | &#x[0]\*3C6[;] | &#[0]\*966[;] | HTML 4.0 |
| chi | χ | &chi[;] | &#x[0]\*3C7[;] | &#[0]\*967[;] | HTML 4.0 |
| psi | ψ | &psi[;] | &#x[0]\*3C8[;] | &#[0]\*968[;] | HTML 4.0 |
| omega | ω | &omega[;] | &#x[0]\*3C9[;] | &#[0]\*969[;] | HTML 4.0 |
| thetasym | ϑ | &thetasym[;] | &#x[0]\*3D1[;] | &#[0]\*977[;] | HTML 4.0 |
| upsih | ϒ | &upsih[;] | &#x[0]\*3D2[;] | &#[0]\*978[;] | HTML 4.0 |
| piv | ϖ | &piv[;] | &#x[0]\*3D6 | &#[0]\*982[;] | HTML 4.0 |
| ensp |  | &ensp[;] | &#x[0]\*2002[;] | &#[0]\*8194[;] | HTML 4.0 |
| emsp |  | &emsp[;] | &#x[0]\*2003[;] | &#[0]\*8195[;] | HTML 4.0 |
| thinsp |  | &thinsp[;] | &#x[0]\*2009[;] | &#[0]\*8201[;] | HTML 4.0 |
| zwnj |  | &zwnj[;] | &#x[0]\*200C[;] | &#[0]\*8204[;] | HTML 4.0 |
| zwj |  | &zwj[;] | &#x[0]\*200D[;] | &#[0]\*8205[;] | HTML 4.0 |
| lrm |  | &lrm[;] | &#x[0]\*200E[;] | &#[0]\*8206[;] | HTML 4.0 |
| rlm |  | &rlm[;] | &#x[0]\*200F[;] | &#[0]\*8207[;] | HTML 4.0 |
| ndash | – | &ndash[;] | &#x[0]\*2013[;] | &#[0]\*8211[;] | HTML 4.0 |
| mdash | — | &mdash[;] | &#x[0]\*2014[;] | &#[0]\*8212[;] | HTML 4.0 |
| lsquo | ‘ | &lsquo[;] | &#x[0]\*2018[;] | &#[0]\*8216[;] | HTML 4.0 |
| rsquo | ’ | &rsquo[;] | &#x[0]\*2019[;] | &#[0]\*8217[;] | HTML 4.0 |
| sbquo | ‚ | &sbquo[;] | &#x[0]\*201A[;] | &#[0]\*8218[;] | HTML 4.0 |
| ldquo | “ | &ldquo[;] | &#x[0]\*201C[;] | &#[0]\*8220[;] | HTML 4.0 |
| rdquo | ” | &rdquo[;] | &#x[0]\*201D[;] | &#[0]\*8221[;] | HTML 4.0 |
| bdquo | „ | &bdquo[;] | &#x[0]\*201E[;] | &#[0]\*8222[;] | HTML 4.0 |
| dagger | † | &dagger[;] | &#x[0]\*2020[;] | &#[0]\*8224[;] | HTML 4.0 |
| Dagger | ‡ | &Dagger[;] | &#x[0]\*2021[;] | &#[0]\*8225[;] | HTML 4.0 |
| bull | • | &bull[;] | &#x[0]\*2022[;] | &#[0]\*8226[;] | HTML 4.0 |
| hellip | … | &hellip[;] | &#x[0]\*2026[;] | &#[0]\*8230[;] | HTML 4.0 |
| permil | ‰ | &permil[;] | &#x[0]\*2030[;] | &#[0]\*8240[;] | HTML 4.0 |
| prime | ′ | &prime[;] | &#x[0]\*2032[;] | &#[0]\*8242[;] | HTML 4.0 |
| Prime | ″ | &Prime[;] | &#x[0]\*2033[;] | &#[0]\*8243[;] | HTML 4.0 |
| lsaquo | ‹ | &lsaquo[;] | &#x[0]\*2039[;] | &#[0]\*8249[;] | HTML 4.0 |
| rsaquo | › | &rsaquo[;] | &#x[0]\*203A[;] | &#[0]\*8250[;] | HTML 4.0 |
| oline | ‾ | &oline[;] | &#x[0]\*203E[;] | &#[0]\*8254[;] | HTML 4.0 |
| frasl | ⁄ | &frasl[;] | &#x[0]\*2044[;] | &#[0]\*8260[;] | HTML 4.0 |
| euro | € | &euro[;] | &#x[0]\*20AC[;] | &#[0]\*8364[;] | HTML 4.0 |
| image | ℑ | &image[;] | &#x[0]\*2111[;] | &#[0]\*8465[;] | HTML 4.0 |
| weierp | ℘ | &weierp[;] | &#x[0]\*2118[;] | &#[0]\*8472[;] | HTML 4.0 |
| real | ℜ | &real[;] | &#x[0]\*211C[;] | &#[0]\*8476[;] | HTML 4.0 |
| trade | ™ | &trade[;] | &#x[0]\*2122[;] | &#[0]\*8482[;] | HTML 4.0 |
| alefsym | ℵ | &alefsym[;] | &#x[0]\*2135[;] | &#[0]\*8501[;] | HTML 4.0 |
| larr | ← | &larr[;] | &#x[0]\*2190[;] | &#[0]\*8592[;] | HTML 4.0 |
| uarr | ↑ | &uarr[;] | &#x[0]\*2191[;] | &#[0]\*8593[;] | HTML 4.0 |
| rarr | → | &rarr[;] | &#x[0]\*2192[;] | &#[0]\*8594[;] | HTML 4.0 |
| darr | ↓ | &darr[;] | &#x[0]\*2193[;] | &#[0]\*8595[;] | HTML 4.0 |
| harr | ↔ | &harr[;] | &#x[0]\*2194[;] | &#[0]\*8596[;] | HTML 4.0 |
| crarr | ↵ | &crarr[;] | &#x[0]\*21B5[;] | &#[0]\*8629[;] | HTML 4.0 |
| lArr | ⇐ | &lArr[;] | &#x[0]\*21D0[;] | &#[0]\*8656[;] | HTML 4.0 |
| uArr | ⇑ | &uArr[;] | &#x[0]\*21D1[;] | &#[0]\*8657[;] | HTML 4.0 |
| rArr | ⇒ | &rArr[;] | &#x[0]\*21D2[;] | &#[0]\*8658[;] | HTML 4.0 |
| dArr | ⇓ | &dArr[;] | &#x[0]\*21D3[;] | &#[0]\*8659[;] | HTML 4.0 |
| hArr | ⇔ | &hArr[;] | &#x[0]\*21D4[;] | &#[0]\*8660[;] | HTML 4.0 |
| forall | ∀ | &forall[;] | &#x[0]\*2200[;] | &#[0]\*8704[;] | HTML 4.0 |
| part | ∂ | &part[;] | &#x[0]\*2202[;] | &#[0]\*8706[;] | HTML 4.0 |
| exist | ∃ | &exist[;] | &#x[0]\*2203[;] | &#[0]\*8707[;] | HTML 4.0 |
| empty | ∅ | &empty[;] | &#x[0]\*2205[;] | &#[0]\*8709[;] | HTML 4.0 |
| nabla | ∇ | &nabla[;] | &#x[0]\*2207[;] | &#[0]\*8711[;] | HTML 4.0 |
| isin | ∈ | &isin[;] | &#x[0]\*2208[;] | &#[0]\*8712[;] | HTML 4.0 |
| notin | ∉ | &notin[;] | &#x[0]\*2209[;] | &#[0]\*8713[;] | HTML 4.0 |
| ni | ∋ | &ni[;] | &#x[0]\*220B[;] | &#[0]\*8715[;] | HTML 4.0 |
| prod | ∏ | &prod[;] | &#x[0]\*220F[;] | &#[0]\*8719[;] | HTML 4.0 |
| sum | ∑ | &sum[;] | &#x[0]\*2211[;] | &#[0]\*8721[;] | HTML 4.0 |
| minus | − | &minus[;] | &#x[0]\*2212[;] | &#[0]\*8722[;] | HTML 4.0 |
| backslash | \ | &backslash[;] | &#x[0]\*2216[;] | &#x[0]\*8726[;] | HTML 5.0 |
| lowast | ∗ | &lowast[;] | &#x[0]\*2217[;] | &#[0]\*8727[;] | HTML 4.0 |
| radic | √ | &radic[;] | &#x[0]\*221A[;] | &#[0]\*8730[;] | HTML 4.0 |
| prop | ∝ | &prop[;] | &#x[0]\*221D[;] | &#[0]\*8733[;] | HTML 4.0 |
| infin | ∞ | &infin[;] | &#x[0]\*221E[;] | &#[0]\*8734[;] | HTML 4.0 |
| ang | ∠ | &ang[;] | &#x[0]\*2220[;] | &#[0]\*8736[;] | HTML 4.0 |
| and | ∧ | &and[;] | &#x[0]\*2227[;] | &#[0]\*8743[;] | HTML 4.0 |
| or | ∨ | &or[;] | &#x[0]\*2228[;] | &#[0]\*8744[;] | HTML 4.0 |
| cap | ∩ | &cap[;] | &#x[0]\*2229[;] | &#[0]\*8745[;] | HTML 4.0 |
| cup | ∪ | &cup[;] | &#x[0]\*222A[;] | &#[0]\*8746[;] | HTML 4.0 |
| int | ∫ | &int[;] | &#x[0]\*222B[;] | &#[0]\*8747[;] | HTML 4.0 |
| there4 | ∴ | &there4[;] | &#x[0]\*2234[;] | &#[0]\*8756[;] | HTML 4.0 |
| sim | ∼ | &sim[;] | &#x[0]\*223C[;] | &#[0]\*8764[;] | HTML 4.0 |
| cong | ≅ | &cong[;] | &#x[0]\*2245[;] | &#[0]\*8773[;] | HTML 4.0 |
| asymp | ≈ | &asymp[;] | &#x[0]\*2248[;] | &#[0]\*8776[;] | HTML 4.0 |
| ne | ≠ | &ne[;] | &#x[0]\*2260[;] | &#[0]\*8800[;] | HTML 4.0 |
| equiv | ≡ | &equiv[;] | &#x[0]\*2261[;] | &#[0]\*8801[;] | HTML 4.0 |
| le | ≤ | &le[;] | &#x[0]\*2264[;] | &#[0]\*8804[;] | HTML 4.0 |
| ge | ≥ | &ge[;] | &#x[0]\*2265[;] | &#[0]\*8805[;] | HTML 4.0 |
| sub | ⊂ | &sub[;] | &#x[0]\*2282[;] | &#[0]\*8834[;] | HTML 4.0 |
| sup | ⊃ | &sup[;] | &#x[0]\*2283[;] | &#[0]\*8835[;] | HTML 4.0 |
| nsub | ⊄ | &nsub[;] | &#x[0]\*2284[;] | &#[0]\*8836[;] | HTML 4.0 |
| sube | ⊆ | &sube[;] | &#x[0]\*2286[;] | &#[0]\*8838[;] | HTML 4.0 |
| supe | ⊇ | &supe[;] | &#x[0]\*2287[;] | &#[0]\*8839[;] | HTML 4.0 |
| oplus | ⊕ | &oplus[;] | &#x[0]\*2295[;] | &#[0]\*8853[;] | HTML 4.0 |
| otimes | ⊗ | &otimes[;] | &#x[0]\*2297[;] | &#[0]\*8855[;] | HTML 4.0 |
| perp | ⊥ | &perp[;] | &#x[0]\*22A5[;] | &#[0]\*8869[;] | HTML 4.0 |
| sdot | ⋅ | &sdot[;] | &#x[0]\*22C5[;] | &#[0]\*8901[;] | HTML 4.0 |
| lceil | ⌈ | &lceil[;] | &#x[0]\*2308[;] | &#[0]\*8968[;] | HTML 4.0 |
| rceil | ⌉ | &rceil[;] | &#x[0]\*2309[;] | &#[0]\*8969[;] | HTML 4.0 |
| lfloor | ⌊ | &lfloor[;] | &#x[0]\*230A[;] | &#[0]\*8970[;] | HTML 4.0 |
| rfloor | ⌋ | &rfloor[;] | &#x[0]\*230B[;] | &#[0]\*8971[;] | HTML 4.0 |
| lang | 〈 | &lang[;] | &#x[0]\*2329[;] | &#[0]\*9001[;] | HTML 4.0 |
| rang | 〉 | &rang[;] | &#x[0]\*232A[;] | &#[0]\*9002[;] | HTML 4.0 |
| loz | ◊ | &loz[;] | &#x[0]\*25CA[;] | &#[0]\*9674[;] | HTML 4.0 |
| spades | ♠ | &spades[;] | &#x[0]\*2660[;] | &#[0]\*9824[;] | HTML 4.0 |
| clubs | ♣ | &clubs[;] | &#x[0]\*2663[;] | &#[0]\*9827[;] | HTML 4.0 |
| hearts | ♥ | &hearts[;] | &#x[0]\*2665[;] | &#[0]\*9829[;] | HTML 4.0 |
| diams | ♦ | &diams[;] | &#x[0]\*2666[;] | &#[0]\*9830[;] | HTML 4.0 |
| plus | + | &plus[;] | &#x[0]\*2B[;] | &#[0]\*43[;] |  |
| sol | / | &sol[;] | &#x[0]\*2F[;] | &#[0]\*47[;] |  |
| ast | \* | &ast[;] | &#x[0]\*2A[;] | &#[0]\*42[;] |  |
| hyphen | - | &hyphen[;] | &#x[0]\*2D[;] | &#[0]\*45[;] |  |
| lcub | { | &lcub[;] | &#x[0]\*7B[;] | &#[0]\*123[;] |  |
| rcub | } | &rcub[;] | &#x[0]\*7D[;] | &#[0]\*125[;] |  |
| equals | = | &equals[;] | &#x[0]\*3D[;] | &#[0]\*61[;] |  |
| commat | @ | &commat[;] | &#x[0]\*40[;] | &#[0]\*64[;] |  |
| semi | ; | &semi[;] | &#x[0]\*3B[;] | &#[0]\*59[;] |  |
| colon | : | &colon; | &#x[0]\*3A[;] | &#[0]\*58[;] |  |
| num | # | &num[;] | &#x[0]\*23[;] | &#[0]\*35[;] |  |
| quest | ? | &quest[;] | &#x[0]\*3F[;] | &#[0]\*63[;] |  |
| excl | ! | &excl[;] | &#x[0]\*21[;] | &#[0]\*33[;] |  |
| period | . | &period[;] | &#x[0]\*2E[;] | &#[0]\*46[;] |  |

Table 1 - "Character entity references in HTML"

## HTML attribute decoder

HTML attribute decoder is the same as HTML decoder. (See section)

## URL decoder

URL decoding, also known as percent-decoding. URL encoding string is used to identify a name or a resource on the Internet. It is known as Uniform Resource Identifier (URI).

The characters allowed in a URI either reserved or unreserved (see and )

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ! | \* | ‘ | ( | ) | ; | : | @ | & | = | + | $ | , | / | ? | # | [ | ] |

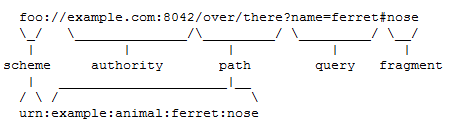
Table 2 - “Reserved characters for URI”

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| R | S | T | U | V | W | X | Y | Z | a | b | c | d | e | f | g | h |
| i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y |
| z | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | - | \_ | . | ~ |  |  |

Table 3 – “Unreserved characters for URI”

The purpose of reserved characters is to provide a set of delimiting characters that are distinguishable from other data within a URI.

For example:



If a reserved character is found in a URI component and no delimiting role is known for that character, then it must be interpreted as representing the data octet corresponding to that character's encoding in ASCII.

Non-ACSII and reserved characters should be converted to a byte sequence in UTF-8. Percent (%) symbol preceded a byte sequence.

For example, the reserved characters should be encoded to follow hexadecimal sequence (see ).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ! | \* | ‘ | ( | ) | ; | : | @ | & | = | + | $ | , | / | ? | # | [ | ] |
| %21 | %2A | %27 | %28 | %29 | %3B | %3A | %40 | %26 | %3D | %2B | %24 | %2C | %2F | %3F | %23 | %5B | %5D |

Table 4 – “Reserved characters for URI”

Common characters after URL encoding contain follow hexadecimal sequence.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| < | > | ~ | . | " | { | } | | | \ | - | ` | \_ | ^ | % | space |
| %3C | %3E | %7E | %2E | %22 | %7B | %7D | %7C | %5C | %2D | %60 | %5F | %5E | %25 | %20 |

Table 5 - "Common character after URL encoding"

For example

URL encoded string:

http%3A%2F%2Fwww.xxx.com%2Fexample.html%3Fword1%3Dtest%26word2%3D%D0%BF%D1%80%D0%B8%D0%B2%D0%B5%D1%82%21+%27%22%5E%24%26

Should be decoded to:

http://www.xxx.com/example.html?word1=test&word2=привет! '"^$&

For translate UTF-8 sequence (which could represented from one to four bytes) to Unicode characters see Appendix A.2 – Unicode characters and UTF-8 .

Also URL decoder must support Unicode decoding

For example

URL encoded Unicode code ‘**%u0430**’ must decoded to Cyrillic letter ‘**A**’

## JavaScript decoder

JavaScript decoder must support:

1) ASCII format;

2) Hexadecimal code of special characters (See ) represented in following format \**xHH**  , where HH is a two hehadecimal digits;

3) Unicode sequences represented in following format \**uHHHH**, where HHHH is a four hexadecimal digits.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Char** | **Hex** |  | **Char** | **Hex** |  | **Char** | **Hex** |  | **Char** | **Hex** |
| space | 20 | ' | 27 | : | 3a | [ | 5b |
| ! | 21 | ( | 28 | ; | 3b | \ | 5c |
| " | 22 | ) | 29 | < | 3c | ] | 5d |
| # | 23 | \* | 2a | = | 3d | ^ | 5e |
| $ | 24 | + | 2b | > | 3e | { | 7b |
| % | 25 |  | , | 2c |  | ? | 3f |  | } | 7d |
| & | 26 |  | / | 2f |  | @ | 40 |  | " | 7e |

Table 6.1 - "Hexadecimal code of special JavaScript characters"

4) JavaScript escape sequences consisting of backslash (\) and following by any other character. Such sequence is treated either as especial character (See Table 6.2 – “ Special JavaScript characters ”) or as a character after backslash. I.e., \\, \n, \t, \”, etc.

|  |  |
| --- | --- |
| **Char** | **Hex** |
| \' | single quote |
| \" | double quote |
| \& | ampersand |
| \\ | backslash |
| \n | new line |
| \r | carriage return |
| \t | tab |
| \b | backspace |
| \f | form feed |

Table 6.2 - "Special JavaScript characters"

5) Octal code **\777** from two to three digits, where 7 – is octal code from 0 to 7. Also if octal code more then 177 (127 in decimal code) the decoder should be decoded only two digits (for more details see example).

For example

JavaScript encoded string:

\x3Cscript\x3Ealert\x28\x22\x20\u041F\u0440\u0438\u0432\u0435\u0442\x21\x20\x2B\x22\x29\x3B

Should be decoded to:

<script>alert(" Привет! +");

Example with octal code.

JavaScript encoded string: \41 Should be decoded to: !

JavaScript encoded string: \717 Should be decoded to: 97 (decoded only two digits. The third digit is left unchanged)

## VBScript decoder

VBScript decoder must support:

1) ASCII format:

* alphabet (a-z or A-Z);
* numbers (0-9);
* dot (.);
* coma (,);
* dash (-);
* underscore (\_);
* space ( );

2) Hexadecimal code of other special characters (See );

3) Unicode sequences.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Char** | **Hex** | **Decimal** |  | **Char** | **Hex** |  |  | **Char** | **Hex** | **Decimal** |  | **Char** | **Hex** | **Decimal** |
| ! | [0]\*21 | [0]\*33 | ( | [0]\*28 | [0]\*40 | < | [0]\*3c | [0]\*60 | ] | [0]\*5d | [0]\*93 |
| " | [0]\*22 | [0]\*34 | ) | [0]\*29 | [0]\*41 | = | [0]\*3d | [0]\*61 | ^ | [0]\*5e | [0]\*94 |
| # | [0]\*23 | [0]\*35 | \* | [0]\*2a | [0]\*42 | > | [0]\*3e | [0]\*62 | { | [0]\*7b | [0]\*123 |
| $ | [0]\*24 | [0]\*36 | + | [0]\*2b | [0]\*43 | ? | [0]\*3f | [0]\*63 | } | [0]\*7d | [0]\*125 |
| % | [0]\*25 | [0]\*37 | / | [0]\*2f | [0]\*47 | @ | [0]\*40 | [0]\*64 | " | [0]\*7e | [0]\*126 |
| & | [0]\*26 | [0]\*38 |  | : | [0]\*3a | [0]\*58 |  | [ | [0]\*5b | [0]\*91 |  |  |  |  |
| ' | [0]\*27 | [0]\*39 |  | ; | [0]\*3b | [0]\*59 |  | \ | [0]\*5c | [0]\*92 |  |  |  |  |

Table 7 - "Hexadecimal code of special VBScript characters"

Special symbols or Unicode sequence represented as hexadecimal or decimal code which the following format:

* hexadecimal format for single-byte character is **chr[[3]](#footnote-3)(&hHH),** where HH is a 2 of hexadecimal digits;
* hexadecimal format for Unicode code point is **chrw[[4]](#footnote-4)(&hHHHH),** where HHHH is a max 4 of hexadecimal digits;
* hexadecimal format for single-byte character is **chr3(DDD),** where DDD is a max 3 of decimal digits;
* decimal format – **chrw4(DDDDD),** where DDDDD is a max 5 of decimal digits;

Encoding symbols or ASCII string concatenated with ampersand (&) symbol.

For example

VBScript encoded string:

“alert"&chrw(40)&chrw(39)&"XSS атака"&chrw(33)&chrw(39)&chrw(41)&chrw(59)&" .,-\_”

Should be decoded to:

alert('XSS атака!'); .,-\_

## XML decoder

XML decoder is the same as HTML decoder. But the XML specification is more strict and defines five "predefined entities" representing special characters. XML decoder should be supported three formats it is a entity string, a hexadecimal and decimal formats:

1. Character entity format **&name;** where name is case-sensitive alphanumeric string;
2. Hexadecimal format, **&#x[0]\*2HH;** where HH is a max 2 of hexadecimal digits;
3. Decimal format, **&#[0]\*DD;** where DD is a max 2 of decimal digits;

Character entity references in XML see

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Character** | **Entity** | **Hex code** | **Decimal code** |
| quot | " | &quot; | &#x[0]\*22; | &#[0]\*34; |
| amp | & | &amp; | &#x[0]\*26; | &#[0]\*38; |
| apos | ' | &apos; | &#x[0]\*27; | &#[0]\*39; |
| lt | < | &lt; | &#x[0]\*3C; | &#[0]\*60; |
| gt | > | &gt; | &#x[0]\*3E; | &#[0]\*62; |

Table 8 - "Character entity references in XML"

Other character can be used in ASCII or Unicode standards.

## XML attribute decoder

XML attribute decoder is the same as XML decoder. (See section)

## LDAP query decoder

Lightweight Directory Access Protocol (LDAP) query decoder decodes sequences with a backslash (\) followed two digits. Five special sequences should decode to ASCII characters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \* | ( | ) | / | \ |
| \2A | \28 | \29 | \2F | \5c |

Table 9 - "ASCII characters after LDAP encoding"

Other UTF-8 string represent as a backslash (\) followed by the two hexadecimal digits representing the value of the encoded octet (For more detail about UTF-8 encoding see Appendix A.2 – Unicode characters and UTF-8 encodings).

For example

LDAP query encoded string:

ldapsearch -x -H ldap:\2f\2fhost:port -LLL -b "cn=\d0\a1\d1\85\d0\b5\d0\bc\d0\b0" -s base '\28objectClass=\2a\29' suppotedControls

Should be decoded to:

ldapsearch -x -H ldap://host:port -LLL -b "cn=Схема" -s base '(objectClass=\*)' suppotedControls

## CSS decoder

Cascading Style Sheet (CSS) decoder decodes sequences with a backslash (\) followed between one and six hexadecimal digits that represent a character code from the ISO 10646 standard (which is equivalent to Unicode, for all intents and purposes). Any character other than a hexadecimal digit will terminate the escape sequence. If a character following the escape sequence is also a valid hexadecimal digit then it must either include six digits in the escape, or use a whitespace character to terminate the escape, if the sequence less the six digits.

For example

CSS encoded string:

\3c span\20 style\3d \22 font\2d family\3a\20LU\20 beck\22 \3e \2e \2e \2e \3c \2f span\3e

Or

\00003Cspan\000020style\00003D\000022font\00002Dfamily\00003A\000020LU\000020beck\000022\00003E\00002E\00002E\00002E\00003C\00002Fspan\00003E

Should be decoded to:

<span style=\"font-family: LU beck\">...</span>

## Base64 decoder

Base64 decoder translates a string which represented a radix-64 format (or Base64 – it’s a similar). Radix-64 – it is the scheme that represented binary data in a ASCII string. Common algorithm to translation into radix-64 format is a representation is base on a 64-character alphabet. The alphabet is shown in

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Char** | **Value** | |  | **Char** | **Value** | |  | **Char** | **Value** | |  | **Char** | **Value** | |
| **Decimal** | **Hex** | **Decimal** | **Hex** | **Decimal** | **Hex** | **Decimal** | **Hex** |
| A | 0 | 00 | Q | 16 | 10 | g | 32 | 20 | w | 48 | 30 |
| B | 1 | 01 | R | 17 | 11 | h | 33 | 21 | x | 49 | 31 |
| C | 2 | 02 | S | 18 | 12 | i | 34 | 22 | y | 50 | 32 |
| D | 3 | 03 | T | 19 | 13 | j | 35 | 23 | z | 51 | 33 |
| E | 4 | 04 | U | 20 | 14 | k | 36 | 24 | 0 | 52 | 34 |
| F | 5 | 05 | V | 21 | 15 | l | 37 | 25 | 1 | 53 | 35 |
| G | 6 | 06 | W | 22 | 16 | m | 38 | 26 | 2 | 54 | 36 |
| H | 7 | 07 | X | 23 | 17 | n | 39 | 27 | 3 | 55 | 37 |
| I | 8 | 08 | Y | 24 | 18 | o | 40 | 28 | 4 | 56 | 38 |
| J | 9 | 09 | Z | 25 | 19 | p | 41 | 29 | 5 | 57 | 38 |
| K | 10 | 0A | a | 26 | 1A | q | 42 | 2A | 6 | 58 | 3A |
| L | 11 | 0B | b | 27 | 1B | r | 43 | 2B | 7 | 59 | 3B |
| M | 12 | 0C | c | 28 | 1C | s | 44 | 2C | 8 | 60 | 3C |
| N | 13 | 0D | d | 29 | 1D | t | 45 | 2D | 9 | 61 | 3D |
| O | 14 | 0E | e | 30 | 1E | u | 46 | 2E | ….. | 62 | 3E |
| P | 15 | 0F | f | 31 | 1F | v | 47 | 2F | …. | 63 | 3F |

Table 10 - "Base64 alphabet characters"

The 62 and 63 characters can be variation and it is depend of implementation.

The length of the input string must be multiple of four. If the first rule is true, then the input sequence must be transformed using the following algorithm. (See and read this scheme upward)

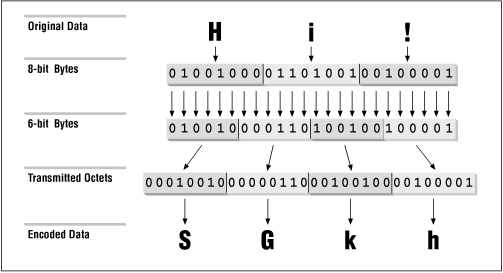


Figure 1 - "Base64 encoding"

Base64 decoder must transforms sequence from various schemes. All Base64 schemes have three differences.

1. Char for index 62;
2. Char for index 63;
3. Pad char.

The table below summarizes these known variants.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Variant** | **Char for index 62** | **Char for index 63** | ***pad* char** | **Fixed encoded line-length** | **Maximum encoded line length** | **Line separators** | **Characters outside alphabet** | **Line checksum** |
| **Original Base64 for**[**Privacy-Enhanced Mail (PEM)**](http://en.wikipedia.org/wiki/Base64#Privacy-enhanced_mail)**(RFC 1421, deprecated)** | + | / | =*(mandatory)* | Yes (except last line) | 64 | CR+LF | Forbidden | *(none)* |
| **Base64 transfer encoding for**[**MIME**](http://en.wikipedia.org/wiki/Base64#MIME)**(RFC 2045)** | + | / | =*(mandatory)* | No (variable) | 76 | CR+LF | Accepted (discarded) | *(none)* |
| **Standard 'Base64' encoding for**[**RFC 3548**](http://en.wikipedia.org/wiki/Base64#RFC_3548)**or**[**RFC 4648**](http://en.wikipedia.org/wiki/Base64#RFC_4648) | + | / | =*(mandatory)* | Yes (except last line) | 64 or 76 *(only if line separators are specified and needed)* | CR+LF *(only if specified and needed)* | Forbidden | *(none)* |
| **'Radix-64' encoding for**[**OpenPGP**](http://en.wikipedia.org/wiki/Base64#OpenPGP)**(RFC 4880)** | + | / | =*(mandatory)* | No (variable) | 76 | CR+LF | Forbidden | 24-bit CRC (Radix-64-encoded, including one *pad* character) |
| **Modified Base64 encoding for**[**UTF-7**](http://en.wikipedia.org/wiki/Base64#UTF-7)**(RFC 1642, obsoleted)** | + | / | *(none)* | No (variable) | *(none)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**filenames**](http://en.wikipedia.org/wiki/Base64#Filenames)**(non standard)** | + | - | *(none)* | No (variable) | *(filesystem limit, generally 255)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**URL applications**](http://en.wikipedia.org/wiki/Base64#URL_applications)**('base64url' encoding)** | - | \_ | *(none)* | No (variable) | *(application-dependent)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**XML**](http://en.wikipedia.org/wiki/Base64#XML)**name tokens (*Nmtoken*)** | . | - | *(none)* | No (variable) | *(XML parser-dependent)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**XML**](http://en.wikipedia.org/wiki/Base64#XML)**identifiers (*Name*)** | \_ | : | *(none)* | No (variable) | *(XML parser-dependent)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**Program identifiers**](http://en.wikipedia.org/wiki/Base64#Program_identifiers)**(variant 1, non standard)** | \_ | - | *(none)* | No (variable) | *(language/system-dependent)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**Program identifiers**](http://en.wikipedia.org/wiki/Base64#Program_identifiers)**(variant 2, non standard)** | . | \_ | *(none)* | No (variable) | *(language/system-dependent)* | *(none)* | Forbidden | *(none)* |
| **Modified Base64 for**[**Regular expressions**](http://en.wikipedia.org/wiki/Base64#Regular_expressions)**(non standard)** | ! | - | *(none)* | No (variable) | *(application-dependent)* | *(none)* | Forbidden | *(none)* |

Table 11 - "Base64 variants"

For example:

Input sequence:

**SGk=**

And it should be decoded to:

**Hi**

## GZip decoder

GZip decoder must uncompress input sequence by GNU zip utility.

End return a byte sequence.

## Null HTML sanitizer

This sanitizer must remove null HTML expression which represented a **\0** sequence.

**\0** sequence is not rendered any Internet browsers, but if was a multiply encoding this sequence can be interpreted as hex code of some hexadecimal sequence.

# Functional Area

| Requirement | Story | Constraints |
| --- | --- | --- |
| R-400 – Mandatory requirement | Create HTM decoder |  |
| R-401 – Mandatory requirement | Create HTML attribute decoder |  |
| R-402 – Mandatory requirement | Create URL decoder |  |
| R-403 – Mandatory requirement | Create JavaScript decoder |  |
| R-404 – Mandatory requirement | Create VBScript decoder |  |
| R-405 – Mandatory requirement | Create XML decoder |  |
| R-406 – Mandatory requirement | Create XML attribute decoder |  |
| R-407 – Mandatory requirement | Create LDAP query decoder |  |
| R-408 – Mandatory requirement | Create CSS decoder |  |
| R-409 – Mandatory requirement | Create Base64 decoder |  |
| R-410 – Mandatory requirement | Create GZip decoder |  |
| O-409 – Mandatory requirement | Create null HTML sanitizer |  |

# Apendix A – Unicode, UTF-16 and UTF-8 encodings

UTF stands for Unicode Transformation Format. The '8' means it uses 8-bit blocks to represent a character. The number of blocks needed to represent a character varies from 1 to 4. The '16' means it uses 16-bit blocks to represent a character. The number of blocks needed to represent a character varies from 1 to 2.

## Appendix A.1 – Unicode characters and UTF-16 encodings

For characters equal to or below 65535 (hex 0xFFFF), the UTF-16 representation is spread across one word (2 bytes or 16 bites).

For characters equal to or greater than 65536 but less that 1114111 (0x10FFFF), the UTF-16 representation is spread across two word (4 bytes or 21 bites). The first byte will have the 8,7,5,4 bites are set and the 6,3,2,1,0 bites are clear (i.e. 0xD800). The third byte will have the 8,7,5,43 bites are set and the 6,,2,1,0 bites are clear (i.e. 0xDC00).

The following table shows the format of such UTF-8 byte sequences (where the "free bits" shown by x's in the table are combined in the order shown, and interpreted from most significant to least significant).

**Binary format of bytes in sequence**

|  |  |  |  |
| --- | --- | --- | --- |
| **1st and 2nd bytes** | **3rd and 4th bytes** | **Number of Free Bits** | **Maximum Expressible Unicode Value** |
| xxxxxxxx xxxxxxxx |  | 16 | FFFF hex (65535) |
| 11011xxx xxxxxxxx | 110111xx xxxxxxxx | (3+8+2+8)=21 | 20000 hex (2097152) |

The value of each individual byte indicates its UTF-8 function, as follows:

* 0000 to FFFF hex (0 to 65535): two bytes.
* 10000 to 10FFF hex (65536 to 69631): four bytes.

## Appendix A.2 – Unicode characters and UTF-8 encodings

Comparison table of Unicode code point and UTF-8 hex code is represent in <http://www.utf8-chartable.de/unicode-utf8-table.pl>.For any character equal to or below 127 (hex 0x7F), the UTF-8 representation is one byte. It is just the lowest 7 bits of the full unicode value. This is also the same as the ASCII value. These characters are represented in the table since U+0000 and finishing U+007F

For characters equal to or below 2047 (hex 0x07FF), the UTF-8 representation is spread across two bytes. The first byte will have the two high bits set and the third bit clear (i.e. 0xC2 to 0xDF). The second byte will have the top bit set and the second bit clear (i.e. 0x80 to 0xBF). These characters are represented in the table since U+0080 and finishing U+07FF.

For characters equal to or greater than 2048 but less that 65535 (0xFFFF), the UTF-8 representation is spread across three bytes. The first byte will have the three high bits set and the forth bit clear (i.e. 0xE0 to 0xEF). These characters are represented in the table since U+0800 and finishing U+FFFF.

For characters equal to or greater than 65536 but less that 1049599 (0x1003FF), the UTF-8 representation is spread across four bytes. The first byte will have the four high bits set and the five bit clear (i.e. 0xF0 to 0xF4). These characters are represented in the table since U+10000 and finishing U+1003FF.

The following table shows the format of such UTF-8 byte sequences (where the "free bits" shown by x's in the table are combined in the order shown, and interpreted from most significant to least significant).

**Binary format of bytes in sequence**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1st Byte** | **2nd Byte** | **3rd Byte** | **4th Byte** | **Number of Free Bits** | **Maximum Expressible Unicode Value** |
| 0xxxxxxx |  |  |  | 7 | 007F hex (127) |
| 110xxxxx | 10xxxxxx |  |  | (5+6)=11 | 07FF hex (2047) |
| 1110xxxx | 10xxxxxx | 10xxxxxx |  | (4+6+6)=16 | FFFF hex (65535) |
| 11110xxx | 10xxxxxx | 10xxxxxx | 10xxxxxx | (3+6+6+6) = 21 | 1003FF hex (1049599) |

The value of each individual byte indicates its UTF-8 function, as follows:

* 00 to 7F hex (0 to 127): first and only byte of a sequence.
* 80 to BF hex (128 to 191): continuing byte in a multi-byte sequence.
* C2 to DF hex (194 to 223): first byte of a two-byte sequence.
* E0 to EF hex (224 to 239): first byte of a three-byte sequence.
* F0 to F4 hex (240 to 244): first byte of a four-byte sequence.

UTF-8 remains a simple, single-byte, ASCII-compatible encoding method, as long as no characters greater than 127 are directly present. This means that an HTML document technically declared to be encoded as UTF-8 can remain a normal single-byte ASCII file. The document can remain so even though it may contain Unicode characters above 127, as long as all characters above 127 are referred to indirectly by ampersand entities.

## Appendix A.3 – Unicode characters in hexadecimal notation

|  |  |  |
| --- | --- | --- |
| **16-bit Unicode** | **UTF-16 Sequence** | **UTF-8 Sequence** |
| U+0001 | 01 | 01 |
| U+007F | 7F | 7F |
| U+0080 | 80 | C2 80 |
| U+07FF | 7FF | DF BF |
| U+800 | 800 | E0 A0 80 |
| U+FFFF | FFFF | EF BF BF |
| U+10000 | D800 DC00 | F0 90 80 80 |
| U+1003FF | DBC0 DFFF | F4 80 8F BF |

1. [] – contains not required symbols, for example [;] – could contains or doesn’t contain a semicolon; [↑](#footnote-ref-1)
2. []\* - contains not required symbols zero or any quantity characters, for example [0]\* - could contains or doesn’t contain any quantity of zero. [↑](#footnote-ref-2)
3. chr - used as ascaped insensitive character for hex or decimal code. [↑](#footnote-ref-3)
4. chrw - used as ascaped insensitive character for hex or decimal code. [↑](#footnote-ref-4)