UASG

Reviewing programming languages and frameworks for compliance with Universal Acceptance good practice

Version 1.1

July 13th, 2018

Table of contents

[**Universal Acceptance**](#_gjdgxs) **5**

[**About this document**](#_cojk4s1xvkbf) **5**

[**Target audience**](#_gj3tf6l2vhfb) **6**

[**Background**](#_dmgc1cbyv1ej) **6**

[**Terminology**](#_cde364h84gus) **6**

[**References**](#_ehqz9zhpikuc) **7**

[**Candidate list of libraries**](#_7hf94c2arih5) **7**

[**Basis of library evaluation**](#_5v7otipetsfo) **8**

[Test suite](#_lg3d3jqyp75m) 8

[Assumptions](#_n2u28dnxgwxl) 9

[4. Technical evaluation](#_r3zg6mxv1ja7) 9

[4.1. Test suite](#_vs91nzf7dta4) 9

[4.1.1 Low-level functions](#_j5bflb4ir3xk) 10

[4.1.1.1 L-U2A: IDNA2008 - Convert Unicode domain name to ASCII lookup form](#_8clf0z9kfea3) 10

[4.1.1.3 L-A2U: IDNA2008 - Convert ASCII domain name to Unicode](#_f0jf28v4w0m) 11

[4.1.2 High-level functions](#_n3cxf45v96hb) 12

[4.1.2.1 H-DNS: Domain name - syntactic check](#_whohy6909qd3) 12

[4.1.2.3 H-ES: Email- syntactic check](#_ovfmprhbxc9q) 13

[4.1.2.6 H-ID: Identifier - Identifier lookup](#_5c0ln9w3gwdg) 14

[**Appendix A - Code examples**](#_npdcbb818z9w) **15**

[1. L-U2A: IDNA2008 - Convert Unicode domain name to ASCII lookup form](#_2mi2g6spqfha) 15

[GNU Libidn2 (C)](#_h0bfob21cpiw) 15

[2. L-A2U: IDNA2008 - Convert ASCII domain name to Unicode](#_tsm1jparf1m) 16

[npm idna-uts46 (Javascript)](#_xqlmw0g9qusn) 16

[**Appendix B - References**](#_gjgoe775mdh3) **17**

[UASG Documents](#_6aeyo27avycx) 17

[IDNA RFCs](#_l3qto0mjh4um) 17

[IANA Registries](#_3brd8oyeqaxe) 17

[Unicode](#_7u2s8dn27mbr) 17

[PRECIS RFC](#_dxzxca835oeg) 17

[Special-use domain name RFCs](#_mshmz1hem0ik) 17

[Internationalized email RFCs](#_fdlutl80fj4b) 18

[Obsolete IDNA RFCs](#_yeij5oady411) 18

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Authors | Notes |
| 1.1 | 13 Jul 2018 | Viagénie | Shortened and updated version. |
| 1.0 | 21 Aug 2017 | Jim Hague  Sara Dickinson | Review call 2017-08-09 updates. |
| 0.99 | 4 Aug 2017 | Jim Hague  Sara Dickinson | Address comments on previous versions. Prepping for review call. |
| 0.98 | 11 Jul 2017 | Jim Hague  Sara Dickinson | Add an Appendix in a separate document to describe the test sets in detail and link to documents containing example test data.  Also some minor updates to the main document. |
| 0.97 | 11 Apr 2017 | Jim Hague  Sara Dickinson  John Dickinson | Incorporate received comments and corrections, review |
| 0.96 | 10 Mar 2017 | Jim Hague  Sara Dickinson | Revised to follow test suite based approach |
| 0.95 | 19 Jan 2017 | Frank Michlick | Updated with feedback after first team review |
| 0.90 | 24 Oct 2016 | Frank Michlick | First version |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 

# 

# 

# 

# Universal Acceptance

Universal Acceptance is a foundational requirement for a truly multilingual Internet, one in which users around the world can navigate entirely in local languages. It is also the key to unlocking the potential of new generic top-level domains (gTLDs) to foster competition, consumer choice and innovation in the domain name industry. To achieve Universal Acceptance, Internet applications and systems must treat all TLDs in a consistent manner, including new gTLDs and internationalized TLDs. Specifically, they must accept, validate, store, process and display all domain names.

The Universal Acceptance Steering Group (UASG) is a community-based team working to share this vision for the Internet of the future with those who construct this space: coders. The group's primary objective is to help software developers and website owners understand how to update their systems to keep pace with an evolving domain name system (DNS).

About this document

This document was created to provide a framework for the evaluation of popular programming packages and libraries and their usefulness in aiding Universal Acceptance good practice. It is a response to the description of work issued by UASG, available at <https://uasg.tech/wp-content/uploads/2016/05/Help-Wanted-Open-Source-Software-Review-v201602111.pdf>.

Where those packages or libraries do not provide the expected support, the follow-up project would create recommendations/patches to be submitted to add UA support and guidance for application developers on correct and effective use of the packages or libraries.

This document and the evaluation process it describes are expected to evolve together as experience is gained with evaluations.

Technical details required by those performing library evaluations are presented ina separate document. This separation of documents is purely due to technical restrictions in the document platform[[1]](#footnote-2).

# Target audience

The main part of this document serves as an overview of the evaluation framework presented in a form appropriate for all stakeholders.

Those performing the library evaluations require the technical details of the tests and the test data sets presented in a separate document and accompanying documents. The test descriptions in this main document describe the categories of tests, omitting technical detail, and are accompanied by illustrative individual tests using data drawn from [UASG004 - Use Cases for UA Readiness Evaluation](https://uasg.tech/wp-content/uploads/2017/05/UASG004-Use-Cases-for-UA-Readiness-Evaluation-2017-04-17.pdf).

# Background

Software applications that make use of Internet services are built and used in a variety of ways. They exist at all points along a continuum ranging from embedded firmware in a connected device, through desktop/mobile/tablet applications, through to software that runs purely in a web browser environment, the latter often communicating with more software running on remote servers.

All these types make use of Internet identifiers which, while historically represented only in characters employed by US English (i.e. A-Z, 0-9 and ‘-’), can now, via the IDNA Protocol, be fully multilingual. These identifiers are:

* Domain names, e.g. example.com or 普遍接受-测试.世界
* Email addresses, e.g. joe.bloggs@example.com or 测试3@普遍接受-测试.top

It is therefore important for all stakeholders in development of a software application to be aware what libraries are available for their chosen development environment to be used for processing Internet identifiers, and to have a clear basis for assessing those libraries, for technical and business suitability, with regard to the UA correctness and compliance.

# Terminology

**Libraries**: All but the most unusual Internet applications today rely heavily on software components to perform much of their function. These components are variously termed packages, frameworks or libraries (which may or may not include various bindings); for brevity, they will all be referred to as **libraries** henceforth.

**Functions**: Similarly, the services offered by these libraries may be variously classed as methods, functions, APIs etc. but will be referred to simply as **functions** henceforth.

**Identifier**: Any of the Internet identifiers: domain names, email address.

References

A full list of references is given in [Appendix B](#_gjgoe775mdh3).

The illustrative test data presented in the main document is drawn from

[UASG004 - Use Cases for UA Readiness Evaluation](https://uasg.tech/wp-content/uploads/2017/05/UASG004-Use-Cases-for-UA-Readiness-Evaluation-2017-04-17.pdf). Other references list the relevant standards and related information in the following categories:

* IDNA RFCs
* Unicode
* IANA Registries
* Special-use domain name RFCs
* Internationalized email RFCs
* Obsolete IDNA RFCs

# Candidate list of libraries

This section lists a currently popular set of libraries that are good candidates for an initial set of evaluations.

* [GNU Libidn](https://www.gnu.org/software/libidn). Implementation of IDNA2003 in C. Bindings available for Perl and Ruby.
* [GNU Libidn2](https://www.gnu.org/software/libidn/" \l "libidn2). Implementation of IDNA2008 in C by the author of GNU Libidn.
* [International Components for Unicode](http://site.icu-project.org/). Versions are available for Java and for C with C and C++ bindings.
* [Python encodings.idna](https://docs.python.org/3/library/codecs.html" \l "module-encodings.idna). Part of the Python standard library. Test in Python and Python3.
* [Python idna module](https://pypi.python.org/pypi/idna). A replacement for the Python standard library encodings.idna module that supports IDNA2008. Test in Python and Python3.
* [PHP IDN functions](http://php.net/manual/en/ref.intl.idn.php). Part of the PHP standard library, supporting IDNA2003 and IDNA2008.
* [Go idna package](https://godoc.org/golang.org/x/net/idna). Part of the Go standard library supporting IDNA2008.
* [Javascript idna-uts46 npm module](https://www.npmjs.com/package/idna-uts46). Supports IDNA2003 and IDNA2008. Bundled with Node.js.

The evaluation process will initially consider only Open Source libraries.

Basis of library evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Test suite

This document outlines the test cases to be used in the test suite.

The scope of this document is to specify tests that provide specific evaluation criteria as a starting point for an evaluation of the overall library quality for typical use cases[[2]](#footnote-4). A test suite providing comprehensive test coverage for all functions is large task[[3]](#footnote-5), and is outside the scope of the current statement of work.

As noted, many libraries are available with bindings enabling them to be used by languages other than their implementing language. In this case, the test suite must be based on using the binding language.

It is strongly recommended that test suites generated as a result of these evaluations should be published under an Open Source license, to aid the growth over time of a comprehensive test resource.

To aid in evaluation, we classify library functions into two groups:

* **Low level functions**: those that provide basic lower-level services, typically transformations defined in the IDNA RFCs.
* **High level functions**: those directed at higher-level application tasks such as syntactic and semantic checks. These will typically include calls to lower-level functions to perform their tasks.

This classification allows the assessor and potential library user to judge whether functionality provided by a particular library is likely to be sufficient in and of itself, or whether further application code or other libraries will be necessary. It is highly preferable that higher level functions are implemented by libraries to avoid multiple application developers having to separately reproduce this functionality which would likely result in errors and inconsistencies.

## Assumptions

We also make some basic assumptions about the programming languages being employed.

* Unicode strings may be represented and manipulated by commonly available facilities. While this is universally true for all popular contemporary languages, the internal encoding a language will use to represent Unicode varies between languages; UTF-8 and UTF-16 are popular choices. When a Unicode string is required for a test, it is assumed it will be represented in the encoding native for the language.
* The language may make use of any available operating system services, including the network stack.

These are both reasonable assumptions for all modern programming environments, and also serve to prevent the scope of this document expanding unreasonably.

In other words, this framework is not designed for evaluating heavily restricted or specialised languages and runtimes.

# 4. Technical evaluation

### 4.1. Test suite

The test suite is designed to evaluate compliance with behaviour described in RFCs and other standards. The great advantage of standards based evaluations is the relative ease with which compliance can measured and the general interoperability that results.

The way in which libraries present services varies significantly between libraries. It should not be assumed that the functions given below are matched by a single function in the library under test. Rather, they are an attempt to describe function at a high enough level that results from a test suite created for a particular library are comparable with results from other libraries.

Evaluators are encouraged to consider if the test suite can be written in a way that aligns with the current test framework of the library under adoption. Evaluators can then easily make library maintainers aware of the tests and maximise the chances of future adoption of the tests by the library maintainers.

A summary of the test cases is given below including a test case ID which is used later in the document. A more detailed description, notes and illustrative test data follow.

**Low-level functions (see section 4.1.1)**

1. **L-U2A:** IDNA2008 - Convert Unicode domain name to ASCII lookup form
2. **L-A2U:** IDNA2008 - Convert ASCII domain name to Unicode

For libraries that only support IDNA2003, the tests should be implemented using the available IDNA2003 functions. Having only IDNA2003 support will give rise to test failures.

**High level functions (see section 4.1.2)**

1. **H-DNS:** Domain name - syntactic check
2. **H-ES:** Email address - syntactic check
3. **H-ID**: Identifiant - Identifiant lookup

[Appendix A](#_ufkyvwas5ppe) provides some code samples for example test cases from the test suite.

**Test cases**

|  |
| --- |
| The full technical description of the proposed test cases is in a separate document**.** Each test case has an input description, expected result, test purpose and standards reference. The document describes multiple test cases, each of which should have at least one piece of test data, preferably multiple where applicable. *It is intended as a technical reference for evaluators.* |

|  |
| --- |
|  |

**Example test data**

The descriptions of the tests below are accompanied by a small number of example test data items with brief descriptions, largely drawn from [UASG0004](https://uasg.tech/wp-content/uploads/2017/05/UASG004-Use-Cases-for-UA-Readiness-Evaluation-2017-04-17.pdf). *These are for illustrative purposes only and cover a small subset of the test cases in the full technical description.*

4.1.1 Low-level functions

Low-level functions provide basic transformation services required by the relevant IDNA RFCs.

##### 4.1.1.1 L-U2A: IDNA2008 - Convert Unicode domain name to ASCII lookup form

*Scenario*: Convert a domain name in Unicode to ASCII using the process described in RFC5891 for domain name lookup. If the domain name, or any constituent label, is already in ASCII, the ASCII should not be altered.

*References*: RFC5891, UTS#46

*Sample test data*:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected output** | **Comment** |
| ua-test.link | ua-test.link | Verify ASCII passed unaltered. |
| 普遍接受-测试.top | xn----f38am99bqvcd5liy1cxsg.top | Verify subdomain conversion. |
| ua-test.世界 | ua-test.xn--rhqv96g | Verify TLD conversion. |
| 普遍接受-测试.世界 | xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g | Verify all-Unicode conversion. |
| 普遍接受-测试。世界 | xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g | Verify Open Dot is recognised as label separator. |
| ua-test.xn--rhqv96g | ua-test.xn--rhqv96g | Verify ACE encoded TLD is passed as ASCII. |
| xn----f38am99bqvcd5liy1cxsg.top | xn----f38am99bqvcd5liy1cxsg.top | Verify ACE encoded subdomain is passed as ASCII. |
| xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g | xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g | Verify all-ACE encoded domain is passed as ASCII. |
| fußballplatz.de | xn--fuballplatz-w6a.de (non-transitional) | Verify IDNA2008 processing. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

##### 4.1.1.3 L-A2U: IDNA2008 - Convert ASCII domain name to Unicode

*Scenario*: Convert a domain name in ASCII to Unicode using the process described in RFC5891. If the domain name, or any constituent label, is already in Unicode or an ASCII label does not begin with the ACE prefix, the original label should not be altered.

*References*: RFC5891, RFC3492

*Sample test data*:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected output** | **Comment** |
| ua-test.link | ua-test.link | Verify ASCII passed unaltered. |
| xn----f38am99bqvcd5liy1cxsg.top | 普遍接受-测试.top | Verify Unicode conversion in subdomain. |
| ua-test.xn--rhqv96g | ua-test.世界 | Verify Unicode conversion in TLD. |
| xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g | 普遍接受-测试.世界 | Verify all-Unicode conversion. |
| xn--fuballplatz-w6a.de | fußballplatz.de | Verify IDNA2008. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

#### 4.1.2 High-level functions

High-level functions reflect operations that an application is likely to want to perform but which are not directly detailed in an IDNA RFC, rather they either build on the operations specified therein or on separate RFCs relating to identifiers.

High-level functions provide basic syntactic checks and decomposition functions described by the relevant RFCs for the identifier. For the purposes of this document, we define a *syntactic check* as a check that a value obeys the rules of form (typically defined in a RFC) for that identifier. In other words, that the value is a potentially valid. So, for example, for a value to be a syntactically valid domain name it must pass all the rules laid down in the relevant RFCs for a domain name - overall length and individual label lengths must be within the prescribed limits, it must not contain any disallowed code points etc.

These functions may be provided directly by the library or implemented with standard library functions.

##### 4.1.2.1 H-DNS: Domain name - syntactic check

*Scenario*: Perform a syntactic check on a domain name. Determine whether the name appears to be correctly formed. If any part of the name already appears to be in ASCII form (an A-label), verify it can be converted to Unicode.

*References*: RFC5891, RFC1035, SAC053

*Sample test data*:

|  |  |  |
| --- | --- | --- |
| **Name** | **Syntactically correct?** | **Comment** |
| ua-test.link | Yes | Verify ASCII. |
| xn----f38am99bqvcd5liy1cxsg.TOP | Yes | Verify ACE plus ASCII. |
| 普遍接受-测试.top | Yes | Verify Unicode subdomain. |
| ua-test.世界 | Yes | Verify Unicode TLD. |
| ua-test.invalid | Yes | Verify non-existent domain,to ensure check is purely syntactic. |
| ua-test..invalid | No | Verify empty label prohibited. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

4.1.2.3 H-ES: Email- syntactic check

*Scenario*: Perform a syntactic check on an email address. Determine whether the address appears to be correctly formed.

*References*: RFC5891, RFC6531

*Sample test data*:

|  |  |  |
| --- | --- | --- |
| **Name** | **Syntactically correct?** | **Comment** |
| info@ua-test.link | Yes | Verify ASCII. |
| info@普遍接受-测试.top | Yes | Verify ASCII with Unicode subdomain. |
| info@普遍接受-测试.世界 | Yes | Verify ASCII mailbox, Unicode domain. |
| données@ua-test.link | Yes | Verify Unicode mailbox, ASCII domain. |
| info@ua-test.invalid | Yes | Verify non-existent domain. |
| info@@ua-test.technology | No | Verify single @. |
| info@ua-test..technology | No | Verify empty label disallowed. |

4.1.2.6 H-ID: Identifier - Identifier lookup

##### *Scenario*: Compare the identifier stored in the system against the one used to authenticate by the user. The test cases below aims to validate proper handling of internationalized identifiers by applications.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*References*: RFC8264

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Sample test data*:

|  |  |  |  |
| --- | --- | --- | --- |
| **Registration username** | **Authentication username** | **Matches?** | **Comment** |
| user | user | Yes | Verify ASCII. |
| identité | identité | Yes | Verify Unicode |
| identité (é = U+00E9) | identité  (é = U+0065 U+0301) | Yes | Verify Unicode normalization |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 

# Appendix A - Code examples

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |

## 

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

In practice, the low-level and high-level functions above are unlikely to be implemented in a consistent fashion across different libraries. This appendix gives examples of how some might be implemented as part of the test case using different libraries.

## 1. L-U2A: IDNA2008 - Convert Unicode domain name to ASCII lookup form

### GNU Libidn2 (C)

#include <locale.h>

#include <stdio.h>

#include <stdlib.h>

#include <idn2.h>

int main(int ac, char \*av[])

{

const char \*name = u8"普遍接受-测试.世界";

int rc;

char \*lookupname;

setlocale(LC\_ALL, "");

rc = idn2\_lookup\_ul(name, &lookupname, 0);

if ( rc != IDN2\_OK )

{

fprintf(stderr,

"error: %s (%s, %d)\n",

idn2\_strerror(rc),

idn2\_strerror\_name(rc),

rc);

return 1;

}

printf("DNS lookup of %s: %s\n", name, lookupname);

free(lookupname);

return 0;

}

$ ./a.out

DNS lookup of 普遍接受-测试.世界: xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g

## 2. L-A2U: IDNA2008 - Convert ASCII domain name to Unicode

### npm idna-uts46 (Javascript)

'use strict';

var uts46 = require('idna-uts46');

var ascii = "xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g";

var unicode = uts46.toUnicode(ascii);

console.log("DNS " + ascii + ": " + unicode);

$ js example.js

DNS xn----f38am99bqvcd5liy1cxsg.xn--rhqv96g: 普遍接受-测试.世界

# 

# Appendix B - References

# 

# 

UASG Documents

[UASG004 - Use Cases for UA Readiness Evaluation](https://uasg.tech/wp-content/uploads/2017/05/UASG004-Use-Cases-for-UA-Readiness-Evaluation-2017-04-17.pdf)

## IDNA RFCs

[RFC3492 - Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)](https://www.rfc-editor.org/info/rfc3492)

[RFC5890 - Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework](https://www.rfc-editor.org/info/rfc5890)

[RFC5891 - Internationalized Domain Names in Applications (IDNA): Protocol](https://www.rfc-editor.org/info/rfc5891)

[RFC5892 - The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)](https://www.rfc-editor.org/info/rfc5892)

[RFC5893 - Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)](https://www.rfc-editor.org/info/rfc5893)

[RFC5894 - Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale](https://www.rfc-editor.org/info/rfc5894)

## IANA Registries

[IDNA Parameters](https://www.iana.org/assignments/idna-tables-6.3.0/idna-tables-6.3.0.xhtml) (IDNA Contextual Rules and Derived Properties)

## Unicode

[Unicode Technical Standard #46 - Unicode IDNA Compatibility Processing](http://www.unicode.org/reports/tr46/" \l "ProcessingStepNormalize)

## PRECIS RFC

[PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols](https://www.rfc-editor.org/info/rfc8264)

Special-use domain name RFCs

Some domain names are reserved for special use; that is, their use requires special handling at some point in the name resolution process. A full list of these names is given in [IANA list of special-use domain names](https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml).

[RFC6761 - Special-use domain names](https://www.rfc-editor.org/info/rfc6761)

[RFC6762 - Multicast DNS](https://www.rfc-editor.org/info/rfc6762)

[RFC7686 - The ".onion" special-use domain name](https://www.rfc-editor.org/info/rfc7686)

Internationalized email RFCs

[RFC6530 - Overview and Framework for Internationalized Email](https://www.rfc-editor.org/info/rfc6530)

[RFC6531 - SMTP Extension for Internationalized Email](https://www.rfc-editor.org/info/rfc6531)

Obsolete IDNA RFCs

These describe IDNA2003. They are kept here for reference when dealing with libraries that only support IDNA2003.

[RFC3490 - Internationalizing Domain Names in Applications (IDNA)](https://www.rfc-editor.org/info/rfc3490)

[RFC3491 - Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)](https://www.rfc-editor.org/info/rfc3491)

1. The tables in the technical presentation are wide, and best presented in landscape form. Google Docs cannot at present mix portrait and landscape pages in a single document. [↑](#footnote-ref-2)
2. By typical, we mean use cases relating to widely used writing systems. We do not, for example, consider ensuring that domain names in Ancient Egyptian hieroglyphics are correctly handled to be an immediate priority at this moment. [↑](#footnote-ref-4)
3. The Unicode consortium provide [a set of comprehensive test data](ftp://ftp.unicode.org/Public/idna/latest/IdnaMappingTable.txt) for UTS#46 processing. This contains over 7700 test data items. [↑](#footnote-ref-5)