Abstract

Many Internet application protocols include string-based lookup, searching, or sorting operations. However the problem space for searching and sorting international strings is large, not fully explored, and is outside the area of expertise for the Internet Engineering Task Force (IETF). Rather than attempt to solve such a
large problem, this specification creates an abstraction framework so that application protocols can precisely identify a comparison function and the repertoire of comparison functions can be extended in the future.

Table of Contents

1. Introduction .................................................. 4
   1.1. Conventions Used in this Document ...................... 4
2. Collation Definition and Purpose ............................. 4
   2.1. Definition .................................................. 4
   2.2. Purpose .................................................... 4
   2.3. Some Other Terms Used in this Document .................. 5
   2.4. Sort Keys ................................................... 5
3. Collation Identifier Syntax .................................. 6
   3.1. Basic Syntax ............................................... 6
   3.2. Wildcards .................................................. 6
   3.3. Ordering Direction ......................................... 6
   3.4. URIs ....................................................... 7
   3.5. Naming Guidelines ......................................... 7
4. Collation Specification Requirements .......................... 8
   4.1. Collation/Server Interface ................................. 8
   4.2. Operations Supported ...................................... 8
      4.2.1. Validity ................................................. 8
      4.2.2. Equality ............................................... 9
      4.2.3. Substring .............................................. 9
      4.2.4. Ordering .............................................. 10
   4.3. Sort Keys .................................................. 10
   4.4. Use of Lookup Tables .................................... 10
5. Application Protocol Requirements ............................. 11
   5.1. Character Encoding ......................................... 11
   5.2. Operations ................................................ 11
   5.3. Wildcards .................................................. 12
   5.4. Canonicalization Function ................................ 12
   5.5. Disconnected Clients ..................................... 12
   5.6. Error Codes .............................................. 12
   5.7. Octet Collation ........................................... 13
6. Use by Existing Protocols ..................................... 13
7. Collation Registration ......................................... 13
   7.1. Collation Registration Procedure ........................ 13
   7.2. Collation Registration Format ............................. 14
      7.2.1. Registration Template ................................. 14
      7.2.2. The collation Element ................................. 15
      7.2.3. The identifier Element ................................ 15
      7.2.4. The title Element ..................................... 15
      7.2.5. The operations Element ............................... 15
      7.2.6. The specification Element ............................ 15
7.2.7. The submitter Element ........................................ 16
7.2.8. The owner Element ............................................ 16
7.2.9. The version Element .......................................... 16
7.2.10. The variable Element ...................................... 16
7.2.11. The name Element ........................................... 16
7.2.12. The default Element ....................................... 16
7.2.13. The value Element .......................................... 17
7.3. Structure of Collation Registry .............................. 17
7.4. Example Initial Registry Summary ............................. 18
8. Guidelines for Expert Reviewer .................................... 18
9. Initial Collations .................................................. 19
   9.1. ASCII Numeric Collation ...................................... 19
   9.1.1. ASCII Numeric Collation Description .................... 19
   9.1.2. ASCII Numeric Collation Registration ................... 20
   9.2. ASCII Casemap Collation .................................... 20
   9.2.1. ASCII Casemap Collation Description .................... 20
   9.2.2. ASCII Casemap Collation Registration ................... 21
   9.3. Nameprep Collation .......................................... 21
   9.3.1. Nameprep Collation Description .......................... 21
   9.3.2. Nameprep Collation Registration ......................... 22
   9.4. Octet Collation ............................................. 22
   9.4.1. Octet Collation Description .............................. 22
   9.4.2. Octet Collation Registration ............................. 23
10. IANA Considerations .............................................. 23
11. Security Considerations .......................................... 23
12. Acknowledgements ................................................. 23
13. Open Issues ..................................................... 23
14. Change Log ...................................................... 24
   14.1. Changes From -12 .......................................... 24
   14.2. Changes From -11 .......................................... 24
   14.3. Changes From -10 .......................................... 24
   14.4. Changes From -09 .......................................... 24
   14.5. Changes From -08 .......................................... 25
   14.6. Changes From -06 .......................................... 26
   14.7. Changes From -05 .......................................... 26
   14.8. Changes From -04 .......................................... 26
   14.9. Changes From -03 .......................................... 26
   14.10. Changes From -02 ......................................... 27
   14.11. Changes From -01 ......................................... 27
   14.12. Changes From -00 ......................................... 27
15. References ........................................................ 28
   15.1. Normative References ...................................... 28
   15.2. Informative References .................................... 28
Authors’ Addresses .................................................... 30
Intellectual Property and Copyright Statements .................. 31
1. Introduction

The ACAP [12] specification introduced the concept of a comparator (which we call collation in this document), but failed to create an IANA registry. With the introduction of stringprep [6] and the Unicode Collation Algorithm [8], it is now time to create that registry and populate it with some initial values appropriate for an international community. This specification replaces and generalizes the definition of a comparator in ACAP and creates a collation registry.

1.1. Conventions Used in this Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [1].

The attribute syntax specifications use the Augmented Backus-Naur Form (ABNF) [2] notation including the core rules defined in Appendix A. This also inherits ABNF rules from Language Tags [5].

2. Collation Definition and Purpose

2.1. Definition

A collation is a named function which takes two arbitrary length strings as input and can be used to perform one or more of three basic comparison operations: equality test, substring match, and ordering test.

2.2. Purpose

Collations abstraction layer for comparison functions so that these comparison functions can be used in multiple protocols. The details of a particular comparison operation can be specified by someone with appropriate expertise independent of the application protocols that use that collation. This is similar to the way a charset [14] separates the details of octet to character mapping from a protocol specification such as MIME [10] or the way SASL [11] separates the details of an authentication mechanism from a protocol specification such as ACAP [12].
Here is a small diagram to help illustrate the value of this abstraction layer:

```
+-------------------+                         +-----------------+
| IMAP i18n SEARCH  |--+                      | Basic           |
+-------------------+  |                   +--| Collation Spec  |
|                   |  +-----------------+
+-------------------+  |  +-------------+  |  +-----------------+
| ACAP i18n SEARCH  |--+--| Collation   |--+--| A stringprep    |
|  +-------------+  |  | Registry    |  |  | Collation Spec  |
+-------------------+  |  +-----------------+
| ...other protocol |--+                   |  | locale-specific |
+-------------------+                      |  +-----------------+
```

Thus IMAP, ACAP and future application protocols with international search capability simply specify how to interface to the collation registry instead of each protocol specification having to specify all the collations it supports.

2.3. Some Other Terms Used in this Document

The terms client, server and protocol are used in somewhat unusual senses.

Client means a user, or a program acting directly on behalf of a user. This may be an mail reader acting as an IMAP client, or it may be an interactive shell where the user can type protocol directly, or it may be a script or program written by the user.

Server means a program that performs services requested by the client. This may be a traditional server such as an HTTP server, or it may be a Sieve [15] interpreter running a Sieve script written by a user. A server needs to use the operations provided by collations in order to fulfil the client’s requests.

The protocol describes how the client tells the server what it wants done, and (if applicable) how the server tells the client about the results. IMAP is a protocol by this definition, and so is the Sieve language.

2.4. Sort Keys

One component of a collation is a transformation which turns a string into a sort key, which is then used while sorting.

The transformation can range from an identity mapping (e.g., the
i;octet collation (Section 9.4) to a mapping which makes the string unreadable to a human. This is an implementation detail of collations or servers. A protocol SHOULD NOT expose it, since some collations leave the sort key’s format up to the implementation, and current conformant implementations are known to use different formats.

3. Collation Identifier Syntax

3.1. Basic Syntax

The collation identifier itself is a single US-ASCII string beginning with a letter and made up of letters, digits, and one of the following 4 symbols: "-", ";", "=" and ".". The identifier MUST NOT be longer than 254 characters.

\[
\text{collation-char} = \text{ALPHA} / \text{DIGIT} / "-" / ";" / "=" / "."
\]

\[
\text{collation-id} = \text{ALPHA} *253\text{collation-char}
\]

The identifier "default" is reserved. For protocol which have a default collation, "default" refers to that collation. For other protocols, the identifier "default" matches no collations, and servers SHOULD treat it in the same way as they treat nonexistent collations.

3.2. Wildcards

The string a client uses to select a collation MAY contain one or more wildcard ("*"") character which matches zero or more collation-chars. Wildcard characters MUST NOT be adjacent. If the wildcard string matches multiple collations, the server SHOULD select the collation with the broadest scope (preferably international scope), the most recent table versions and the greatest number of supported operations.

\[
\text{collation-wild} = ("*" / (\text{ALPHA} ["*"])) * (\text{collation-char} ["*"])
\]

; MUST NOT exceed 254 characters total

3.3. Ordering Direction

When used as a protocol element for ordering, the collation identifier MAY be prefixed by either "+" or "-" to explicitly specify an ordering direction. "+" has no effect on the ordering operation, while "-" inverts the result of the ordering operation. In general, collation-order is used when a client requests a collation, and
collation-selected is used when the server informs the client of the selected collation.

\[
\text{collation-selected} = \[+\] / \[-\] \text{collation-id}
\]

\[
\text{collation-order} = \[+\] / \[-\] \text{collation-wild}
\]

### 3.4. URIs

Some protocols are designed to use URIs [4] to refer to collations rather than simple tokens. A special section of the IANA web page is reserved for such usage. The "collation-uri" form is used to refer to a specific IANA registry entry for a specific named collation (the collation registration may not actually be present if it is experimental). The "collation-uri" form is an abstract name for an ordering, a collation pattern or a vendor private collator.

\[
\text{collation-uri} = \text{"http://www.iana.org/assignments/collation/" collation-id ".xml"}
\]

\[
\text{collation-uri} = \text{( "http://www.iana.org/assignments/collation/" collation-wild ".xml" ) / other-uri}
\]

\[
\text{other-uri} = <\text{absoluteURI}>; \text{excluding the IANA collation namespace.}
\]

### 3.5. Naming Guidelines

While this specification makes no absolute requirements on the structure of collation identifiers, naming consistency is important, so the following initial guidelines are provided.

Collation identifiers with an international audience typically begin with "i;". Collation identifiers intended for a particular language or locale typically begin with a language tag [5] followed by a "i;". After the first "i;" normally the name of the general collation algorithm, followed by a series of algorithm modifications separated by the "i;" delimiter. Parameterized modifications will use "=" to delimit the parameter from the value. The version numbers of any lookup tables used by the algorithm SHOULD be present as parameterized modifications.

Collation identifiers of the form "*;vnd-domain.com;*" are reserved for vendor-specific collations created by the owner of the domain name following the "vnd-" prefix (e.g. vnd-example.com for the vendor example.com). Registration of such collations (or the name space as a whole) with intended use of "Vendor" is encouraged when a public specification or open-source implementation is available, but is not
4. Collation Specification Requirements

4.1. Collation/Server Interface

The collation itself defines what it operates on. Most collations are expected to operate on character strings. The \texttt{i;octet} (Section 9.4) collation operates on octet strings. The \texttt{i;ascii-numeric} (Section 9.1) operation operates on numbers.

This specification defines the collation interface in terms of octet strings. However, implementations may choose to use character strings instead. Such implementations may not be able to implement e.g. \texttt{i;octet}. Since \texttt{i;octet} is not currently mandatory to implement for any protocol, this should not be a problem.

4.2. Operations Supported

A collation specification MUST state which of the three basic operations are supported (equality, substring, ordering) and how to perform each of the supported operations on any two input character strings including empty strings. Collations must be deterministic, i.e. given a collation with a specific identifier, and any two fixed input strings, the result MUST be the same for the same operation.

In general, collation operations should behave as their names suggest. While a collation may be new, the operations are not, so the new collation’s operations should be similar to those of older collations. For example, a date/time collation should not provide a "substring" operation that would morph IMAP substring SEARCH into e.g. a date-range search.

A nonobvious consequence of the rules for each collation operation is that for any single collation, either none or all of the operations can return "undefined". For example, it is not possible to have an equality operation that never returns "undefined" and a substring operation that occasionally does.

4.2.1. Validity

The validity test takes one string as argument returns valid if its input string is valid input to collation’s other operations, and invalid if not. (In other words, a string is valid if it is equal to itself according to the collation’s equality operation.)

The validity test is provided by all collations. It MUST NOT be
listed separately in the collation registration.

4.2.2. Equality

The equality test always returns "match" or "no-match" when supplied valid input, and MAY return "undefined" if one or both input strings are not valid.

The equality test MUST be reflexive and symmetric. For valid input, it MUST be transitive.

If a collation provides either a substring or an ordering test, it MUST also provide an equality test. The substring and/or ordering tests MUST be consistent with the equality test.

In this specification, the return values of the equality test are called "match", "no-match" and "undefined". This is not a specification, merely a choice of phrasing.

4.2.3. Substring

The substring matching operation determines if the first string is a substring of the second string, i.e. if one or more substrings of the second string is equal to the first, as defined by the collation’s equality operation.

A collation which supports substring matching will automatically support two special cases of substring matching: prefix and suffix matching if those special cases are supported by the application protocol. It returns "match" or "no-match" when supplied valid input and returns "undefined" when supplied invalid input.

Application protocols MAY return position information for substring matches. If this is done, the position information SHOULD include both the starting offset and the ending offset for each match. This is important because more sophisticated collations can match strings of unequal length (for example, a pre-composed accented character can match a decomposed accented character). In general, overlapping matches SHOULD be reported (as when "ana" occurs twice within "banana") although there are cases where a collation may decide not to. For example, in a collation which treats all whitespace sequences as identical, the substring operation could be defined such that " 1 " (SP "1" SP) is reported just once within " 1 " (SP SP "1" SP SP), not four times (SP SP 1 SP, SP 1 SP, SP 1 SP SP and SP SP 1 SP SP).

A string is a substring of itself. The empty string is a substring of all strings.
Note that the substring operation of some collations can match strings of unequal length. For example, a pre-composed accented character can match a decomposed accented character. Unicode Collation Algorithm [8] discusses this in more detail.

In this specification, the return values of the substring operation are called "match", "no-match" and "undefined". This is not a specification, merely a choice of phrasing.

4.2.4. Ordering

The ordering operation determines how two strings are ordered. It MUST be trichotomous and reflexive. For valid input, it MUST be transitive.

Ordering returns "less" if the first string is listed before the second string according to the collation, "greater" if the second string is listed before the first string, and "equal" if the two strings are equal as defined by the collation’s equality operation. If one or both strings are invalid, the result of ordering is "undefined".

When the collation is used with a "+" prefix, the behavior is the same as when used with no prefix. When the collation is used with a "-" prefix, the result of the ordering operation of the collation MUST be reversed.

In this specification, the return values of the ordering operation are called "less", "equal", "greater" and "undefined". This is not a specification, merely a choice of phrasing.

4.3. Sort Keys

A collation specification SHOULD describe the internal transformation algorithm to generate sort keys. This algorithm can be applied to individual strings and the result can be stored to potentially optimize future comparison operations. A collation MAY specify that the sort key is generated by the identity function. The sort key may have no meaning to a human. The sort key may not be valid input to the collation.

4.4. Use of Lookup Tables

Some collations use customizable lookup tables, e.g. because the tables depend on locale and may be modified after shipping the software. Collations which use more than one customizable lookup table in a documented format MUST assign numbers to the tables they use. This permits an application protocol command to access the
tables used by a server collation, so that clients and servers use the same tables.

5. Application Protocol Requirements

This section describes the requirements and issues that an application protocol needs to consider if it offers searching, substring matching and/or sorting, and permits the use of characters outside the US-ASCII charset.

5.1. Character Encoding

The protocol specification has to make sure that it is clear on which characters (rather than just octets) the collations are used. This can be done by specifying the protocol itself in terms of characters (e.g. in the case of a query language), by specifying a single character encoding for the protocol (e.g. UTF-8 [3]), or by carefully describing the relevant issues of character encoding labeling and conversion. In the later case, details to consider include how to handle unknown charsets, any charsets which are mandatory-to-implement, any issues with byte-order that might apply, and any transfer encodings which need to be supported.

5.2. Operations

The protocol must specify which of the operations defined in this specification (equality matching, substring matching and ordering) can be invoked in the protocol, and how they are invoked. There may be more than one way to invoke an operation.

The protocol MUST provide a mechanism for the client to select the collation to use with equality matching, substring matching and ordering.

If a protocol needs a total ordering and the collation chosen does not provide it because the ordering operation returns "undefined" at least once, the recommended fallback is to sort all invalid strings after the valid ones, and use i;octet to order the invalid strings.

Although the collation’s substring function provides a list of matches, a protocol need not provide all that to the client. It may provide only the first matching substring, or even just the information that the substring search matched.

If the protocol provides positional information for the results of a substring match, that positional information SHOULD fully specify the substring(s) in the result that matches independent of the length of
the search string. For example, returning both the starting and ending offset of the match would suffice, as would the starting offset and a length. Returning just the starting offset is not acceptable. This rule is necessary because advanced collations can treat strings of different lengths as equal (for example, pre- composed and decomposed accented characters).

5.3. Wildcards

The protocol MUST specify whether it allows the use of wildcards in collation identifiers or not. If the protocol allows wildcards, then:

The protocol MUST specify how comparisons behave in the absence of explicit collation negotiation or when a collation of "*" is requested. The protocol MAY specify that the default collation used in such circumstances is sensitive to server configuration. The protocol SHOULD provide a way to list available collations matching a given wildcard pattern or patterns.

5.4. Canonicalization Function

If the protocol uses a canonicalization function for strings, then use of collations MAY be appropriate for that function. As an example, many protocols use case independent strings. In most cases, a simple ASCII mapping to upper/lower case works well, as i;ascii-casemap offers. However, in some cases another collation may be better, e.g. to handle Turkish dotted/dotless i. Protocol designers should consider in each case whether to use a specifiable collation.

5.5. Disconnected Clients

If the protocol supports disconnected clients, then a mechanism for the client to precisely replicate the server’s collation algorithm is likely desirable. Thus the protocol MAY wish to provide a command to fetch lookup tables used by charset conversions and collations.

5.6. Error Codes

The protocol specification should consider assigning protocol error codes for the following circumstances:

- The client requests the use of a collation by identifier or pattern, but no implemented collation matches that pattern.
- The client attempts to use a collation for an operation that is not supported by that collation. For example, attempting to use the "i;ascii-numeric" collation for substring matching.
- The client uses an equality or substring matching collation and the result is an error. It may be appropriate to distinguish between the two input strings, particularly when one is supplied
by the client and one is stored by the server. It might also be appropriate to distinguish the specific case of an invalid UTF-8 string.

5.7. Octet Collation

The i;octet (Section 9.4) collation is only usable with protocols based on octet-strings. Clients and servers MUST NOT use i;octet with other protocols.

If the protocol permits the use of collations with data structures other than strings, the protocol MUST describe the default behavior for a collation with those data structures.

6. Use by Existing Protocols

Both ACAP [12] and Sieve [15] are standards track specifications which used collations prior to the creation of this specification and registry. Those standards do not meet all the application protocol requirements described in Section 5.

These protocols allow the use of the i;octet (Section 9.4) collation working directly on UTF-8 data as used in these protocols.

In Sieve, all matches are either true and false. Accordingly, Sieve servers must treat "undefined" and "no-match" results of the equality and substring operations as false, and only "match" as true.

In ACAP and Sieve, there are no invalid strings. In this document’s terms, invalid strings sort after valid strings.

IMAP [16] also collates, although that is explicit only when the COMPARATOR [18] extension is used. The built-in IMAP substring operation and the ordering provided by the SORT [17] extension may not meet the requirements made in this document.

Other protocols may be in a similar position.

In IMAP, the default collation is i;ascii-casemap, because its operations most closely resembles IMAP’s built-in operations.

7. Collation Registration

7.1. Collation Registration Procedure

The IETF will create a mailing list, collation@ietf.org, which can be
used for public discussion of collation proposals prior to registration. Use of the mailing list is strongly encouraged. The IESG will appoint a designated expert who will monitor the collation@ietf.org mailing list and review registrations.

The registration procedure begins when a completed registration template is sent to iana@iana.org and collation@ietf.org. The designated expert is expected to tell IANA and the submitter of the registration within two weeks whether the registration is approved, approved with minor changes, or rejected with cause. When a registration is rejected with cause, it can be re-submitted if the concerns listed in the cause are addressed. Decisions made by the designated expert can be appealed to IESG Applications Area Director, then to the IESG. They follow the normal appeals procedure for IESG decisions.

Collation registrations in a standards track, BCP or IESG-approved experimental RFC are owned by the IETF, and changes to the registration follow normal procedures for updating such documents. Collation registrations in other RFCs are owned by the RFC author(s). Other collation registrations are owned by the individual(s) listed in the contact field of the registration and IANA will preserve this information. Changes to a registration MUST be approved by the owner. In the event the owner cannot be contacted for a period of one month and a change is deemed necessary, the IESG MAY re-assign ownership to an appropriate party.

7.2. Collation Registration Format

Registration of a collation is done by sending a well-formed XML document to collation@ietf.org and iana@iana.org.

7.2.1. Registration Template

Here is a template for the registration:

```xml
<?xml version='1.0'?>
<!DOCTYPE collation SYSTEM 'collationreg.dtd'>
<collation rfc="YYYY" scope="i18n" intendedUse="common">
  <identifier>collation identifier</identifier>
  <title>technical title for collation</title>
  <operations>equality order substring</operations>
  <specification>specification reference</specification>
  <owner>email address of owner or IETF</owner>
  <submitter>email address of submitter</submitter>
  <version>1</version>
</collation>
```
7.2.2. The collation Element

The root of the registration document MUST be a <collation> element. The collation element contains the other elements in the registration, which are described in the following sub-subsections, in the order given here.

The <collation> element MAY include an "rfc=" attribute if the specification is in an RFC. The "rfc=" attribute gives only the number of the RFC, without any prefix, such as "RFC", or suffix, such as ".txt".

The <collation> element MUST include a "scope=" attribute, which MUST have one of the values "i18n", "local" or "other".

The <collation> element MUST include an "intendedUse=" attribute, which must have one of the values "common", "limited", "vendor", or "deprecated". Collation specifications intended for "common" use are expected to reference standards from standards bodies with significant experience dealing with the details of international character sets.

Be aware that future revisions of this specification may add additional function types, as well as additional XML attributes, values and elements. Any system which automatically parses these XML documents MUST take this into account to preserve future compatibility.

7.2.3. The identifier Element

The <identifier> element gives the precise identifier of the collation, e.g. i;ascii-casemap. The <identifier> element is mandatory.

7.2.4. The title Element

The <title> element gives the title of the collation. The <title> element is mandatory.

7.2.5. The operations Element

The <operations> element lists which of the three operations ("equality", "order" or "substring") the collation provides, separated by single spaces. The <operations> element is mandatory.

7.2.6. The specification Element

The <specification> element describes where to find the
specification. The <specification> element is mandatory. It MAY have a URI attribute. There may be more than one <specification> elements, in which case they together form the specification.

If it is discovered that parts of a collation specification conflict, a new revision of the collation is necessary, and the collation@ietf.org mailing list should be notified.

7.2.7. The submitter Element

The <submitter> element provides an RFC 2822 [13] email address for the person who submitted the registration. It is optional if the <owner> element contains an email address.

There may be more than one <submitter> element.

7.2.8. The owner Element

The <owner> element contains either the four letters "IETF" or an email address of the owner of the registration. The <owner> element is mandatory. There may be more than one <owner> element. If so, all owners are equal. Each owner can speak for all.

7.2.9. The version Element

The <version> element is included when the registration is likely to be revised or has been revised in such a way that the results change for certain input strings. The <version> element is optional.

7.2.10. The variable Element

The <variable> element specifies an optional variable using which the collation’s behaviour can be tailored. The <variable> element is optional. When it is used, it must contain <name> and <default> elements and may contain one or more <value> elements.

7.2.11. The name Element

The <name> element specifies the name value of a variable. The <name> element is mandatory.

7.2.12. The default Element

The <default> element specifies the default value of a variable. The <default> element is mandatory.
7.2.13. The value Element

The <value> element specifies a legal value of a variable. The <value> element is optional. If one or more <value> elements are present, only those values are legal. If none is, then the variable’s legal values do not form an enumerated set, and the rules MUST be specified in an RFC accompanying the registration.

7.3. Structure of Collation Registry

Once the registration is approved, IANA will store each XML registration document in a URL of the form

http://www.iana.org/assignments/collation/collation-id.xml

where collation-id is the contents of the identifier element in the registration. Both the submitter and the designated expert are responsible for verifying that the XML is well-formed. The registration document should avoid using new elements. If any are necessary, it is important to be consistent with other registrations.

IANA will also maintain a text summary of the registry under the name

http://www.iana.org/assignments/collation/summary.txt

This summary is divided into four sections. The first section is for collations intended for common use. This section is intended for collation registrations published in IESG approved RFCs or for locally scoped collations from the primary standards body for that locale. The designated expert is encouraged to reject collation registrations with an intended use of "common" if the expert believes it should be "limited", as it is desirable to keep the number of "common" registrations small and high quality. The second section is reserved for limited use collations. The third section is reserved for registered vendor specific collations. The final section is reserved for deprecated collations.
7.4. Example Initial Registry Summary

The following is an example of how IANA might structure the initial registry summary.txt file:

<table>
<thead>
<tr>
<th>Collation</th>
<th>Functions</th>
<th>Scope</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Use Collations:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i;nameprep;v=1;uv=3.2</td>
<td>e, o, s</td>
<td>i18n</td>
<td>[RFC XXXX]</td>
</tr>
<tr>
<td>i;ascii-casemap</td>
<td>e, o, s</td>
<td>Local</td>
<td>[RFC XXXX]</td>
</tr>
<tr>
<td>Limited Use Collations:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i;octet</td>
<td>e, o, s</td>
<td>Other</td>
<td>[RFC XXXX]</td>
</tr>
<tr>
<td>i;ascii-numeric</td>
<td>e, o</td>
<td>Other</td>
<td>[RFC XXXX]</td>
</tr>
<tr>
<td>Vendor Collations:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deprecated Collations:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References
-------


8. Guidelines for Expert Reviewer

The expert reviewer appointed by the IESG has fairly broad latitude for this registry. While a number of collations are expected (particularly customizations of the basic collation for localized use), an explosion of collations (particularly common use collations) is not desirable for widespread interoperability. However, it is important for the expert reviewer to provide cause when rejecting a registration, and when possible to describe corrective action to permit the registration to proceed. The following table includes some example reasons to reject a registration with cause:

- The registration is not a well-formed XML document.
- The registration has an intended use of "common", but there is no evidence the collation will be widely deployed, so it should be listed as "limited".
- The registration has an intended use of "common", but it is redundant with the functionality of a previously registered "common" collation.
- The registration has an intended use of "common", but the specification is not detailed enough to allow interoperable implementations by others.
The collation identifier fails to precisely identify the version numbers of relevant tables to use.

- The registration fails to meet one of the "MUST" requirements in Section 4.

- The collation identifier fails to meet the syntax in Section 3.

- The collation specification referenced in the registration is vague or has optional features without a clear behavior specified.

- The referenced specification does not adequately address security considerations specific to that collation.

- The registration's operations are needlessly different from those of traditional operations.

- The registration's XML is needlessly different from that of already registered collations.

9. Initial Collations

This section describes an initial set of collations for the collation registry.

9.1. ASCII Numeric Collation

9.1.1. ASCII Numeric Collation Description

The "i;ascii-numeric" collation is a simple collation intended for use with arbitrary sized unsigned decimal integer numbers stored as octet strings. US-ASCII digits (0x30 to 0x39) represent digits of the numbers. Before converting from string to integer, the input string is truncated at the first non-digit character. All input is valid; strings which do not start with a digit represent positive infinity.

The collation supports equality and ordering, but does not support the substring operation.

The equality operation returns "match" if the two strings represent the same number (ie. leading zeroes and trailing nondigits are disregarded) and "no-match" if the two strings represent different numbers.

The ordering operation returns "less" if the first string represents a smaller number than the second, "equal" if they represent the same number, and "greater" if the first string represents a larger number than the second.

Some examples: "0" is less than "1", and "1" is less than "4294967298". "4294967298", "04294967298" and "4294967298b" are all equal. "04294967298" is less than ".", ".", "x" and "y" are equal.
9.1.2. ASCII Numeric Collation Registration

<?xml version='1.0'?>
<!DOCTYPE collation SYSTEM 'collationreg.dtd'>
collation rfc="XXXX" scope="other" intendedUse="limited">
  <identifier>i;ascii-numeric</identifier>
  <title>ASCII Numeric</title>
  <operations>equality order</operations>
  <specification>RFC XXXX</specification>
  <owner>IETF</owner>
  <submitter>chris.newman@sun.com</submitter>
</collation>

9.2. ASCII Casemap Collation

9.2.1. ASCII Casemap Collation Description

The "i;ascii-casemap" collation is a simple collation which operates
on octet strings and treats US-ASCII letters case-insensitively. It
provides equality, substring and ordering operations. All input is
valid.

Its equality, ordering and substring operations are as for i;octet,
except that first, the lower-case letters (octet values 97-122) in
each input string are changed to upper case (octet values 65-90).

Care should be taken when using OS-supplied functions to implement
this collation as it is not locale sensitive. Functions such as
strcasecmp and toupper are sometimes locale sensitive and may
inappropriately map lower-case letters other than a-z to upper case.

The i;ascii-casemap collation is well suited to to use with many
internet protocols and computer languages. Use with natural language
is often inappropriate: even though the collation apparently supports
languages such as Italian and English, in real-world use it tends to
stumble over words such as "naive", names such as "Llwyd", people and
place names containing non-ASCII, euro and pound sterling symbols,
quotation marks, dashes/hyphens, etc.
9.2.2. ASCII Casemap Collation Registration

<?xml version='1.0'?>
<!DOCTYPE collation SYSTEM 'collationreg.dtd'>
<collation rfc="XXXX" scope="local" intendedUse="common">
  <identifier>i;ascii-casemap</identifier>
  <title>ASCII Casemap</title>
  <operations>equality order substring</operations>
  <specification>RFC XXXX</specification>
  <owner>IETF</owner>
  <submitter>chris.newman@sun.com</submitter>
</collation>

9.3. Nameprep Collation

9.3.1. Nameprep Collation Description

The "i;nameprep;v=1;uv=3.2" collation is an implementation of the nameprep [7] specification based on normalization tables from Unicode version 3.2. This collation applies the nameprep canonicalization function to both input strings and then returns the result of the i;octet collation on the canonicalized strings. While this collation offers all three operations, the ordering operation it provides is inadequate for use by the majority of the world.

Version number 1 is applied to nameprep as specified in RFC 3491. If the nameprep specification is revised without any changes that would produce different results when given the same pair of input octet strings, then the version number need not be changed.

The table numbers for tables used by nameprep are as follows:

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UnicodeData-3.2.0.txt</td>
</tr>
<tr>
<td>2</td>
<td>Table B.1</td>
</tr>
<tr>
<td>3</td>
<td>Table B.2</td>
</tr>
<tr>
<td>4</td>
<td>Table C.1.2</td>
</tr>
<tr>
<td>5</td>
<td>Table C.2.2</td>
</tr>
<tr>
<td>6</td>
<td>Table C.3</td>
</tr>
<tr>
<td>7</td>
<td>Table C.4</td>
</tr>
<tr>
<td>8</td>
<td>Table C.5</td>
</tr>
<tr>
<td>9</td>
<td>Table C.6</td>
</tr>
<tr>
<td>10</td>
<td>Table C.7</td>
</tr>
<tr>
<td>11</td>
<td>Table C.8</td>
</tr>
<tr>
<td>12</td>
<td>Table C.9</td>
</tr>
</tbody>
</table>
9.3.2. Nameprep Collation Registration

```xml
<?xml version='1.0'?><!DOCTYPE collation SYSTEM 'collationreg.dtd'>
<collation rfc="XXXX" scope="i18n" intendedUse="common">
  <identifier>i;nameprep;v=1;uv=3.2</identifier>
  <title>Nameprep</title>
  <operations>equality order substring</operations>
  <specification>RFC XXXX</specification>
  <owner>IETF</owner>
  <submitter>chris.newman@sun.com</submitter>
  <version>1</version>
</collation>
```

9.4. Octet Collation

9.4.1. Octet Collation Description

The "i;octet" collation is a simple and fast collation intended for use on binary octet strings rather than on character data. Protocols that want to make this collation available have to do so by explicitly allowing it. If not explicitly allowed, it MUST NOT be used. It never returns an "undefined" result. It provides equality, substring and ordering operations.

The ordering algorithm is as follows:
1. If both strings are the empty string, return the result "equal".
2. If the first string is empty and the second is not, return the result "less".
3. If the second string is empty and the first is not, return the result "greater".
4. If both strings begin with the same octet value, remove the first octet from both strings and repeat this algorithm from step 1.
5. If the unsigned value (0 to 255) of the first octet of the first string is less than the unsigned value of the first octet of the second string, then return "less".
6. If this step is reached, return "greater".

This algorithm is roughly equivalent to the C library function memcmp with appropriate length checks added.

The matching operation returns "match" if the sorting algorithm would return "equal". Otherwise the matching operation returns "no-match".

The substring operation returns "match" if the first string is the empty string, or if there exists a substring of the second string of length equal to the length of the first string which would result in a "match" result from the equality function. Otherwise the substring
9.4.2. Octet Collation Registration

This collation is defined with intendedUse="limited" because it can only be used by protocols that explicitly allow it.

```xml
<?xml version='1.0'?>
<!DOCTYPE collation SYSTEM 'collationreg.dtd'>
<collation rfc="XXXX" scope="i18n" intendedUse="limited">
  <identifier>i;octet</identifier>
  <title>Octet</title>
  <operations>equality order substring</operations>
  <specification>RFC XXXX</specification>
  <owner>IETF</owner>
  <submitter>chris.newman@sun.com</submitter>
</collation>
```

10. IANA Considerations

Section 7 defines how to register collations with IANA. Section 9 defines a list of predefined collations, which should be registered when this document is approved and published as an RFC.

11. Security Considerations

Collations will normally be used with UTF-8 strings. Thus the security considerations for UTF-8 [3], stringprep [6] and Unicode TR-36 [9] also apply and are normative to this specification.

12. Acknowledgements

The authors want to thank all who have contributed to this document, including at least John Cowan, Dave Cridland, Mark Davis, Lisa Dusseault, Frank Ellermann, Philip Guenther, Tony Hansen, Kjetil Torgrim Homme, Michael Kay, Alexey Melnikov, Jim Melton and Abhijit Menon-Sen.

13. Open Issues

When converting this to an RFC, several things must be done: Martin Duerst’s name request, checking for unfortunate page breaks, adding a note to the RFC editor to possibly replace the 3066 reference, checking the SP SP "1" SP SP string for correctness.
14. Change Log

14.1. Changes From -12
1. Remove i;basic, to publish it as a separate RFC. Many documents
   are held up by this document, and this document is only help up
   by i;basic.
2. Get rid of all the typoes I could find.
3. Specifically note that the "same" substring match need not always
   be returned in each of its guises.

14.2. Changes From -11
1. Remove the DTD. Permit well-considered extension of the XML.
   Enable the designated expert to block registrations due to
   inappropriate or overly aggressive extension.
2. Rename collation names to collation identifiers. Having both
   names and titles wasn’t good.
3. Removed some open issues after trying to edit, and deciding that
   the existing text was good.
4. Note that in Sieve, invalid strings sort after valid ones.
5. Make i;ascii-numeric as in RFC2244. The task of this document is
   to establish the registry, not change existing collations.

14.3. Changes From -10
1. Updated contact details for Martin Duerst.
2. Various textual improvements.
3. The registration’s file name now has a mandatory .xml extension.
4. Removed binding MUST for Sieve; it’s more appropriate to put that
   in 3028bis.
5. Syntax fix in registration example.
6. When there are multiple specifications, they now act in concert,
   so it’s possible to have e.g. a main specification and multiple
   locale-specific supplements. It is not possible to name multiple
   locations for the same specification any more. That’ll return as
   a comment feature.
7. Hopefully clearer exposition of i;ascii-casemap.
8. The ban on registering octet-based collations is lifted. One
   hopes that the collation mailing list will present a suitable
   threshold - not too high, not too low.
9. The DTD is published where IE can see it while looking at the
   registrations.

14.4. Changes From -09
1. Rename "error" to "undefined", as suggested by Mark Davis. The
   new name makes for nicer prose IMO.
2. 7b=7 according to i;ascii-numeric. ACAP/Sieve need it.
3. Clarified that even though the collation specification returns a
   list of substrings, the protocol/server need not use all of that
   information. (As indeed IMAP SEARCH does not.)
4. Registrations go directly to the collation list _and_ to the
   IANA, not to the IANA and from there forwarded to designated
   expert.
5. Added an acknowledgements list and populated it with a quick grep
   from my mailbox and memory. Surely incomplete.
6. Noted that in sieve, "no-match" and "undefined" must be treated
   in the same way by the engine.
7. Finish the rename from canonical to sort key.
8. Don’t fall back to i;octet from any other collation. Return
   undefined instead. Note that protocols may fall back to i;octet
   to provide total ordering, if necessary.
9. Call the things operations everywhere, not operators/operations.

14.5. Changes From -08
1.  i;ascii-casemap instead of en;ascii-casemap.
2. UCA v 14. Changing to "latest version of UCA" was suggested,
   but rejected since IETF standards reference stable
   specifications, and "latest" is a moving target.
3. Removed all text on multi-valued attributes. Can be added once
   there is a concrete need for it, either in an update to this
   document or in the protocol that needs it.
4. "Collations MUST specify the canonicalization". Well, the UCA
   doesn’t, so I changed that to a MAY.
5. Add some text explaining why one might want to download tables.
6. Changed the remaining instances of "canonicalization" to talk
   about sort keys. Added a note that a collation’s sort key need
   not be valid input to the same collation.
7. Reserve the word "default" and use it to name a protocol’s
   default collation, provided that protocol has a default
   collation. In earlier versions of the draft, "*" was used to
   name the default collation, but "*" also was implicitly defined
   as the most general collation available.
8. Reinstate the different-length example of substring match.
   Explain what an overlapping match is, by the canonical example.
9. Avoid the word "contain" when talking about substring matches.
   Fewer terms is better.
10. Until -07, both a collation and equality/substring/sort was
    called functions. In -07, the trio was renamed as operations.
    Now, the DTD is updated to match.
11. Appeals go to the Apps AD before the general AD, as suggested by
    Spencer Dawkins.
14.6. Changes From -06
1. Clarified equality and identity: equality is as defined by a
collation, identity is stronger.
2. Added reference to
http://www.unicode.org/reports/tr10/#Searching.
3. Don’t describe sort keys as a canonical representation of the
string.
4. Permit disconnected clients to use wildcards. (A disconnected
client has to resolve the wildcard itself, in the same way that a
server would.)
5. Change collation-wild to have the same length limit as collation.
6. Change to use "less" instead of "-1", etc., and specify that it’s
just phrasing, not specification.
7. Don’t describe the equality, substring and ordering operations as
functions. The definition of collation uses the word function
about the collation itself. A function that has three functions?
Something has to give.
8. Strike a requirement that selecting '*' is the same as not
selecting any collation. It restricted the protocol’s default
too much. Existing code wasn’t listening.
9. Left out the canonicalization/sort keys.

14.7. Changes From -05
1. Added definitions of client, server and protocol, and prose to
specify that while the IANA registrations of collations are
written in terms octet strings, implementations may do it
differently.
2. Changed the wording for ascii-numeric to treat the numbers as
numbers, etc.
3. Added explicit property requirements for the three functions,
e.g. that equality be symmetric. Added requirements that the
three functions be consistent, and that if any operations are
present, equality must be (needed for consistency).
4. Random editing, e.g. changing ‘numbers’ for ascii-numeric to
‘integer numbers’.
5. Gave IMAP/SORT/COMPARATOR the same grandfather treatment as ACAP
and SIEVE.

14.8. Changes From -04

Grammar and clarity changes only. One (weak) example added. No
substantive changes.

14.9. Changes From -03

(This does not include all changes made.)
1. Checked and resolved most issues marked ‘check whether this is true’ or similar.
2. Resolved nameprep issue: No.
3. Removed NULL for compatibility with existing collations (IMAP SORT, Sieve).
4. There can be multiple owners and submitters. Say how.
5. Added a requirement that common collations must now be interoperable. Insufficiently detailed specs cannot be "common".
6. Added a guideline that the operations provided by new collations should be reminiscent of similar operations on existing collations.

14.10. Changes From -02

1. Changed from data being octet sequences (in UTF-8) to data being character sequences (with octet collation as an exception).
2. Made XML format description much more structured.
3. Changed <submittor> to <submitter>, because this spelling is much more common.
4. Defined ‘protocol’ to include query languages.
5. Reorganized document, in particular IANA considerations section (which newly is just a list of pointers).
6. Added subsections, and a ‘Structure of this Document’ section.
7. Updated references.
8. Created a ‘Change Log’ chapter, with sections for each draft.
9. Reduced ‘Open issues’ section, open issues are now maintained at http://www.w3.org/2004/08/ietf-collation.

14.11. Changes From -01

Add IANA comment to open issues. Otherwise this is just a re-publish to keep the document alive.

14.12. Changes From -00

1. Replaced the term comparator with collation. While comparator is somewhat more precise because these abstract functions are used for matching as well as ordering, collation is the term used by other parts of the industry. Thus I have changed the name to collation for consistency.
2. Remove all modifiers to the basic collation except for the customization and the match rules. The other behavior modifications can be specified in a customization of the collation.
3. Use ";" instead of "-" as delimiter between parameters to make names more URL-ish.
5. Switched registration template to use XML document.
6. Added a number of useful registration template elements related to the Unicode Collation Algorithm.
7. Switched language from "custom" to "tailor" to match UCA language for tailoring of the collation algorithm.

15. References
15.1. Normative References


15.2. Informative References


RFC 2222, October 1997.


Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.