OSF CHARACTER AND CODE SET REGISTRY

1. INTRODUCTION

In order to promote code set interoperability among OSF technologies and to provide a mechanism for preventing data loss during automated code set conversions, OSF plans to create a simple Character and Code Set Registry. This document describes the aspects of the registry including:

(a) OSF-registered character sets. A ''character set'' is a group of characters without any associated encoding. Examples include the English alphabet, Japanese Kanji, and the characters needed to write European languages. OSF will define and register character sets with approximate repertoires.

(b) Code set registry procedures. A ''code set'' (also called a ''coded character set'') is a mapping of the members of a character set to specific numeric code values. Examples include ASCII, ISO 8859-1 (Latin-1), JIS X0208 (Japanese Kanji). OSF will allow organizations to register code sets. It also will be possible to register a specific implementation of an ''encoding method.'' An encoding method provides rules for combining multiple code sets in a data stream, while an implementation is a specific instantiation of those rules. Examples of encoding method implementations are AJEC (Advanced Japanese EUC Code) or a specific version of Taiwanese Big 5.[1]

(c) Registry maintenance and distribution.

(d) Conversion between local code set string names and registered values.

1.1. Registry Goals

These are the goals of the registry:

(a) Provide unique identification of registered sets. Each registered value will have one and only one meaning.

(b) Create a registry that is easy to use and maintain. The procedure for registering code and character sets and for obtaining information about them should be simple and unbureaucratic. In addition, since OSF has limited resources, the registry must be easy to maintain.

(c) Provide mechanism for evaluating code set compatibility. For those that want it, provide a way to evaluate whether a conversion between pairs of code sets is likely to result in significant data loss.

To meet these goals, this proposal contains simple and straightforward rules and procedures. The rules are designed to meet requirements for the vast majority of code sets. However, there are bound to be some code sets or registration requests that the rules do not cover. OSF will try to be flexible and use common sense in handling such requests.

1.2. Background Information

In creating a registry, OSF is providing a mechanism for promoting code set interoperability, and making it possible for a client to determine whether it can send data to a server with little or no data loss. Programs that need to send or receive code set information use the registered values and can be assured that other OSF technology licensees are using the same values. Assume OSF defines 0x00010001 as being the registered value for ISO 8859-1 (Latin-1), and that a client sends that value to a server. Both client and server can accurately determine the code set, regardless of its local name (`ISO 8859-1, Latin-1, 88591, iso8859-1`, etc.).

---

1. To simplify registry rules, unless otherwise noted, this document uses the single term ''code set'' to refer either to a specific code set or to an encoding method implementation.

While OSF defines the registered values that clients and servers can exchange, OSF does not guarantee that either can handle a registered code set. Suppose the registry contains a value for IBM's pc850 and a client sends that registered value to a server. The server must be capable of recognizing the registered value for pc850. When attempting to map that value to the local name for pc850, the server must not map the value to any other local code set. Whether or not the server has a local name for pc850, or has a converter for it, however, is implementation-defined.

Other RFCs cover code set interoperability and data loss in detail. [RFC 23] describes the mechanisms that are planned for DCE R1.1 in order to provide code set interoperability, and also covers several code set conversion models. [RFC 27] addresses the issue of potential data loss during code set conversions. The latter RFC also introduces the idea of a code set registry and explains how it might be used. Instead of repeating that information here, this document assumes you are familiar with the registry's rationale and uses, and concentrates on the structure and mechanics of that registry.

2. REGISTERED CHARACTER SETS

As part of the mechanism for preventing data loss, OSF registers a group of character sets and describes the approximate repertoire of each. An unsigned16 identifies each character set. (An unsigned16 provides thousands more values than OSF ever expects to use, but the next smaller size — byte, with 256 possible values — is not enough to allow future expansion.)

2.1. Models for Defining Character Sets

The goal of defining character sets is to provide a mechanism for determining whether two different code sets are compatible. Suppose a Japanese SJIS client needs to connect with a server. If the server is running ISO 8859-1, the client probably wants to reject it because the two code sets encode very different character sets and a conversion from SJIS to ISO 8859-1 probably would lose a significant amount of data. However, if the server is running Japanese AJEC, the client probably accepts the connection.

OSF considered three options for handling character sets within this registry:

(a) Define and register character sets with very specific repertoires. Require that any code set claiming to encode a given character set must include every character in that set's repertoire.

(b) Ignore character sets and register code sets only.

(c) Define and register character sets with approximate repertoires. Allow code sets to claim that they encode a given character set if they support all or most of the characters in that set's repertoire.

In the first model, OSF would create and register character sets with specific repertoires. Code sets that include all characters in a given repertoire would be said to encode that character set. If a code set failed to encode even a single character from a repertoire list, it could not be defined as encoding that character set; instead a new character set repertoire might need to be created to cover the code set.

Consider a simple example. Suppose OSF defined the character set Latin-1 and said it contained exactly the repertoire in ISO 8859-1. Further suppose there were a code set `foo` that included every character in ISO 8859-1 except the generic currency symbol. Under the first model, `foo` would not be considered as encoding Latin-1; OSF would need to create and register a separate character set with the Latin-1-minus-currency-symbol repertoire.

Since two or more code sets only rarely encode exactly the same character set, if OSF registered specific repertoires, it would have nearly as many character sets as code sets. There then would be no simple way to determine which code sets really are compatible enough to allow mostly lossless conversions.

In the second model, OSF would omit character sets from its registry and only provide code set registration. Some argued that OSF could meet its goal of providing a mechanism for avoiding data loss if processes compared lists of available converter modules rather than comparing character sets. For example, if a client was using the `foo` code set, and a server was using `bar`, an application could check whether there was a `foo`-to-`bar` converter. If so, it would assume the two sets were compatible; if not, they weren't. The problem with this approach is that it is common for direct converter modules not to exist between compatible code sets. IBM's pc850 and HP's ROMAN8 encode the basic Latin-1 character set, but most systems do not have converters between these two sets. However, it would be possible to convert from one to the other by using the universal set ISO 10646 as an intermediate interchange form.

In the third model, OSF would create general character sets and approximately define their repertoires rather than providing specific lists of characters in the sets. In this ‘‘fuzzy equality’’ model, code sets are considered to encode the same character set if they include most of the characters in the repertoire. Consider the Latin-1 character set. Many computer vendors have code sets for the languages that Latin-1 covers, but the contents of their code sets do not exactly match what is in ISO 8859-1. In the third model, these small deviations are ignored and the slightly differing code sets are considered to encode the Latin-1 character set; they are fuzzily equal.

A fuzzy equality model cannot guarantee completely lossless data conversions, but it does allow a simple method for evaluating general code set compatibility. It therefore meets conversion requirements for most everyday applications. Those applications that require lossless conversions cannot use a fuzzy equality model. They either must require that a client/server connection involves processes using the same code set, or they must add their own routines to guarantee lossless conversions.

2.2. OSF's Choice: Fuzzy Equality

After analyzing the three models for handling character sets in a registry, OSF chose the third option: fuzzy equality. This model meets the registry goals of providing a mechanism for evaluating code set compatibility and also keeping the registry simple. In making this decision, we recognize that the first model allows more precise evaluation, but we felt it was too complicated to implement, use, and maintain.

We also recognize that choosing fuzzy equality means many questions are open to interpretation. For example, this paper defines the Latin-1 character set as containing approximately the characters in ISO 8859-1. If another code set contains all but 10 of these same characters, is that close enough to say it encodes Latin-1? Probably yes, but the answer depends in part on what 10 characters are different. To take an extreme example, if the code set omits all the a's-with-diacritics that are in Latin-1, it probably would not be considered to encode Latin-1. Taking a more realistic example, if it omits some low-use symbols in Latin-1 but includes all the alphabetics, it probably would be considered to encode Latin-1.

2.3. Currently Defined Character Sets

OSF will use member input in defining character sets and their membership. This section lists the OSF-defined character sets and explains the integer values that identify each set.

Here is the current list of OSF-defined character sets:

```
Identifier  Descriptive Name   Approx. Repertoire
----------  ----------------   ------------------
0x0000      /* not used */
0x0001      PCS                /* see below */
0x0011      Latin-1            ISO 8859-1
0x0012      Latin-2            ISO 8859-2
0x0013      Latin-3            ISO 8859-3
0x0014      Latin-4            ISO 8859-4
0x0015      Cyrillic           ISO 8859-5
0x0016      Arabic             ISO 8859-6
0x0017      Greek              ISO 8859-7
0x0018      Hebrew             ISO 8859-8
0x0019      Latin-5            ISO 8859-9
0x001a      /* reserved for future use */
[ . . . ]
0x0050      European           ISO 6937
[ . . . ]
0x0080      Japanese1          JIS X0201
0x0081      Japanese2          JIS X0208
0x0082      Japanese3          JIS X0212
[ . . . ]
0x0100      Korean1            KS C5601
```

```
0x0101      Korean2            KS C5657
[ . . . ]
0x0180      Taiwanese1         CNS 11643 (1986)
0x0181      Taiwanese2         CNS 11643 (1992)
[ . . . ]
0x0200      Thai               TIS 620-2529
[ . . . ]
0x0280      Indian             LTD 37(1610)
[ . . . ]
0x1000      Universal          ISO 10646
[ . . . ]
0xf000-     /* reserved for vendor-
  0xffff       or user-defined values */
```

Since standard definitions of character sets do not exist, OSF typically adds an entry if there is an ISO or commonly used national code set for a given region or script. That's why, for example, the Latin-3 and Latin-4 character sets appear on this list, even though they are relatively rarely used.

The numbering scheme groups character sets into more-or-less logical, 128-member (hex 0x7f) blocks. Those that cover the sets encoded in the ISO 8859 series appear in one logical block, as do other national sets. There's a rather large gap between the national character sets and the number for the Universal set. This allows for further national set additions.

The range 0xf000 through 0xffff is reserved for vendor- or user-defined values. OSF will not assign character set values in this range. This range is not controlled, so it is possible for two or more vendors or users to make assignments to a single value. The way to guard against conflicting assignments is to register values with OSF.

2.4. Portable Character Set

Although OSF has decided to create approximate repertoires for most character sets, the Portable Character Set (PCS) is an exception to that rule. It must contain exactly these 95 characters (including the space character, shown here at the beginning of the last line):

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
 !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

The reason for defining a specific repertoire for the PCS is to ensure a basic level of interoperability and connectivity between nodes in a network.

Note, however, that the PCS defines semantic, rather than visual, characters. That means, for example, that a character with the semantics of <backslash> must be supported, even if in some fonts, the backslash glyph has been replaced with a <Yen> sign, <c-cedilla>, or other glyph. A replacement glyph thus has two sets of semantics — its own and that of the glyph it replaces. If a <Yen> appears in place of the <backslash>, it performs the functions of the Japanese currency symbol and of a <backslash>.

In addition, the fact that two glyphs look similar doesn't necessarily mean they are the same semantic character. For instance, a double-byte 'A' or a Greek ALPHA appear very similar to the portable <A>, but these are *not* the same semantic character as the <A>.

2.5. Additional Character Sets

OSF expects the list of defined-and-registered character sets to grow over time. OSF members or other organizations can request additions to the list (for example, Ethiopian, or Devanagari, or miscellaneous symbols). To request an addition, send an email request to `cs_registry@osf.org`. All information in the request must be in the PCS and encoded in the International Reference Version of ISO 646:1991 (same as ASCII). Here is a sample registration template:

```
Proposed Character Set Name:
Contact:
Description:
```

There are no length restrictions on any portion of a character set registry request. The following describes each template field.

(a) *Proposed Character Set Name.* This might be the name of the language (Japanese, Thai, Farsi) that uses the characters in the set, the name of the script from which the characters come (Devanagari, Latin-n, Hangul), or other descriptive name (Math Symbols, Miscellaneous Symbols).

(b) *Contact.* This is the person OSF should contact if there are any questions. This field must include the person's name and email address.

(c) *General Description of the Set.* This provides general information about the repertoire of characters in the set. The purpose is to provide enough information to enable OSF to decide whether this set needs to be added to the registry. If, for example, you propose adding a Latin-15 character set, describe in general what characters this set contains that the other Latin-*n* sets do not. For sets whose contents are not obvious from the name (for example, Symbols) give more detail about the repertoire.

You may include a complete list of the characters in the proposed set in this section. While there is no requirement that you do so, such information may help provide the rationale for a registry request. Suppose OSF receives a request to register a Symbols set, and it is described as containing ''commonly used graphic symbols.'' This does not provide enough detail for OSF to determine what the proposed set contains, or whether it duplicates an existing set. OSF therefore would reject the registry request (or ask for more information). A detailed repertoire list provides the needed information.

After receiving a request, OSF writes a recommended outcome — acceptance with proposed registered integer value, or rejection with rationale — and sends that recommendation via electronic mail to the Registration Review Committee (RRC). This committee consists of representatives from interested OSF Member companies (one represenative per company) and is designed to help OSF screen out errors, redundant sets, unclear definitions, etc. Committee members have two weeks to review the OSF recommendation and make comments or objections. A lack of response is assumed to signal consent.

OSF reviews RRC comments and attempts to resolve remaining issues. OSF makes the final decision in any disputed registration request.

3. CODE SET REGISTRY

This section covers the rules and procedures for registering code sets with OSF.

3.1. Organization Identification

Each code set will be associated with the organization that owns it (ISO with sets like the 8859 family, JIS with sets like X0208 and X0212, Hewlett-Packard with its proprietary sets, etc.). These organizations will be identified two ways — by a 16-bit integer value, and by a string. Following is a description of each.

(a) *16-bit integer.* This is an OSF-assigned unsigned16 value that uniquely identifies the organization that owns the set. Note that X/Open also plans to use a 16-bit integer to represent organizations, and, of course, it would be easier if X/Open and OSF use the same values. OSF will work with X/Open to try to reach agreement. If agreement is impossible, OSF will make its own assignments.

A few organization integer values are pre-defined. They are (see next list for abbreviations):

```
Code          Organization
----          ------------
0x0000        /* not used */
0x0001        ISO
0x0002        OSF
0x0003        ECMA
0x0004        JIS
```

```
0x0005      KS
0x0006      CNS
0xf000-     /* reserved for
  0xffff       user-defined values */
```

(b) *String.* Organization names sometimes appear as strings within the registry. In those cases, there will be a single string form for each organization. For example, while Digital Equipment Corporation is variously known as ‘‘Digital,’’ ‘‘DEC,’’ ‘‘dec,’’ ‘‘Digital Equip.,’’ or others, only one of these names will be chosen for use within string organization name fields.

A few organization names are pre-defined. They are:

```
Name        Organization
----        ------------
ISO         International Organization for Standardization
OSF         Open Software Foundation
ECMA        European Computer Manufacturers Association
JIS         Japanese Industrial Standard
KS          Korean Standard
CNS         Chinese National Standard (Taiwan)
```

Other organizations must supply OSF with the string name by which they want to be known. The name must be representable in the PCS. It may be an acronym (e.g., ISO, OSF) or full name (Hitachi, Groupe Bull). OSF accepts the organization-chosen name unless it conflicts with a previously registered organization, or is likely to be misinterpreted. For example, if there were a company named Information Business Methods and it requested to use ‘‘IBM’’ as its string name before the trademarked IBM did, OSF would reject that request. In this, or any other case of name conflict, OSF works with an organization to choose an alternate name.

The purpose of consistent string names is to make descriptive sections of the registry easier to read.

3.2. Registration Requests

To register a code set, organizations must send an email request to `cs_registry@osf.org`. All information in the request must be in the PCS and encoded in the International Reference Version of ISO 646:1991 (same as ASCII). Here is a sample registration template:

```
Organization String:
Organization ID:
Short Description:
Char Set ID(s):
Contact:
Ordering Info:
Comments:
```

The following describes each template field.

(a) *Organization String.* This is the string name as defined above. If this is the organization’s first registry request, the name here is considered temporary. OSF checks the name against its list of previously assigned names and either accepts or rejects it. Once OSF approves an organization’s name, the organization uses that name on all subsequent registry requests.

(b) *Organization ID.* This is the unsigned16 value as defined above. If this is the organization’s first registry request, this field must be blank. OSF will assign the ID and send that information back to the organization’s contact person.

(c) *Short Description.* Basic information about the set. OSF assigns an integer value to each registered set (described below), and when it publishes the list of values, the information in this field accompanies each value. The description field is an unstructured 60-byte (59 data bytes plus a terminating NULL) string, and so can contain nearly anything pertinent to the set, but potential bits of information to include are:

(i) Code set name.

(ii) Year adopted or version number (at least one is required if this is a new version of a previously registered set).

(iii) Short description of set.

Although the description must only use characters in the PCS, there is no requirement that it be in English. For non-U.S. companies that support many code sets, it may be difficult to express some code set names in English, so it is permissible to use PCS-based phoneticization of local names. Note, however, that English descriptions are likely to be the most widely understood throughout the world.

(d) *Character Set ID(s).* This is one or more unsigned16 values from the OSF list of registered character sets (see Section 2 above). Send email to `cs_registry@osf.org` if you need the current OSF list. Use colons (:) to separate multiple entries in this field.

Organizations decide which character set or sets most closely matches each code set to be registered. Since OSF has chosen a fuzzy equality model, there is no requirement that character and code sets be a perfect match. For example, HP's ROMAN8 is considered to encode the Latin-1 character set even though it does not contain all the characters in ISO 8859-1. If, however, the code set encodes a character set not on the OSF list, the organization must propose an addition to the OSF list as described earlier in this document.

(e) *Contact.* This is the person OSF should contact if there are any questions. This field must include the person's name and Internet email address.

(f) *Ordering Info.* An address to which registry users can send requests to order more information about the set. OSF will not maintain a full catalog of detailed information about each registered set, but some users may want more information. Each registering organization is free to create its own ordering policies (for example, whether/how much to charge for an order). Filling orders is the sole responsibility of the organization; OSF will not attempt to fill an order if the organization in question fails to do so, or fails to provide the information a user wants.

This field must include a hard-copy address, and may also include an Internet email address.

(g) *Comments.* This includes any miscellaneous information about the set that the organization wants to provide. For a request to register a company-specific version of an encoding method implementation (for example, a version of Japanese SJIS), this field should include the names of the code sets used, the ranges they occupy, the control character sequences that invoke each set (if any), and information about any "special-use" areas (for example, user- or vendor-definable ranges).

There are no length restrictions for the last three fields in a registry request (Contact, Ordering Info, and Comments).

Following are examples of the information an organization might provide. These are just examples! Some organization IDs have not yet been assigned and probably will change.

(a) Organization String: ISO
Organization ID: 0x0001
Short Description: 8859-1:1987, Latin Alphabet No. 1
Char Set ID(s): 0x0011
Contact: ISO
Ordering Info:
International Organization for Standardization
1, Rue de Varembé
Case postale 56
CH-1211 Genève 20
Switzerland

(b) Organization String: IBM
Organization ID: 0x0100
Short Description: pc850, code page 850

Char Set ID(s): 0x0011
Contact: Frank Rojas, fxrojas@nlsarch.austin.ibm.com
Ordering Info:
Code Set Inquiries
IBM
<address_TBD>
Email: fxrojas@nlsarch.austin.ibm.com

(c) Organization String: HP
Organization ID: 0x0101
Short Description: ROMAN8; Western European code set
Char Set ID(s): 0x0011
Contact: Arne Thormodsen, arnet@cup.hp.com
Ordering Info:
Code Set Inquiries
Hewlett-Packard
<address_TBD>
Email: arnet@cup.hp.com

(d) Organization String: JIS
Organization ID: 0x0004
Short Description: AJEC:1991, Advanced Japanese EUC Code
Char Set ID(s): 0x0001:0x0080:0x0081:0x0082
Contact: Foo Bar, foo@jis.co.jp
Ordering Info:
Code Set Inquiries
Japanese Industrial Standard
<address_TBD>
Comments:
AJEC is the standard EUC encoding method for Japan. It includes characters from ASCII, JIS X0208:1990, JIS X0201:1976, JIS X0212:1990.

3.3. Analysis of Requests

After OSF receives a registry request, it makes a limited analysis of the set. This analysis is designed to screen out obviously fake or frivilous sets like a code set for Martian, and to screen out duplicate registration requests. Organizations must provide, if requested, an on-line or hard-copy version of sets they want to register. In general, if a set is in use on an existing system, or is an international or national standard, it is eligible to be registered.

If a code set is an international or national standard, no company can register it as a company-proprietary set. For example, since 8859-1 has been registered as an ISO set, no company ''Foo'' can register it as a ''Foo'' set. If a company-proprietary set becomes an international or national standard after being assigned a value in the OSF registry, OSF creates a new value for that set with the international or national standards body listed as the owning organization. This is the only time a single version of a set is eligible to be registered twice. We don't expect this to occur very often, because it is extremely rare for any international or national body to standardize an existing set without making any changes to it.

After OSF completes its analysis, it writes a recommended outcome — acceptance with proposed registered value, or rejection with rationale — and sends that recommendation via electronic mail to the RRC. See Section 2.5 (Additional Character Sets) for a description of the RRC and its duties. As in the case of character sets, OSF reviews RRC comments and attempts to resolve any remaining issues. OSF makes the final decision in any disputed registration request.

3.4. Assigning Registered Values

If OSF accepts a submitted code set, it assigns a 32-bit integer value to it. The upper 16 bits identify the organization that owns the set as defined earlier. The lower 16 bits identify the code set.

A registered value refers to one specific version of a code set. When a code set is revised, the new version gets a new value in the registry. For example, if 0x00010001 is the registered value for ISO 8859-1:1987 and ISO revises the set in 1995, the existing value does not change. Instead, OSF issues a new value for the 1995 version.

It is up to each organization to decide when it has created a new version of an existing set, and therefore needs to reregister that set. On the surface, this may seem like a simple decision — when an existing set changes in any way, it has a new version — but in practice, the decision can be more problematic. Suppose a single character gets assigned to a previously unused code value in an existing set. Some companies may feel the change is too minor to warrant reregistering. Others may decide that it is important enough to get a new registered value, since there may be interoperability problems with a data stream that includes the new character but is identified as containing the older code set. Apparently, this type of question can spark intense debates within companies. Rather than attempt to referree these debates, OSF requires that each organization work this out.

Some 16-bit code set values are predefined and some are reserved for future use. Reserved values allow for updates to existing registered sets, or for additions to a given series. For example, the following lists predefined values for the ISO 8859 series and reserves some values in case there are additions to the series or revisions to any of its existing sets. There is no technical requirement that related sets have sequential registered numbers; this is just for the convenience of registry readers.

```
Value           Identifies
-----           ----------
0x00010001      ISO 8859-1:1987, Latin Alphabet No. 1
0x00010002      ISO 8859-2:1987, Latin Alphabet No. 2
0x00010003      ISO 8859-3:1988, Latin Alphabet No. 3
0x00010004      ISO 8859-4:1988, Latin Alphabet No. 4
0x00010005      ISO 8859-5:1988, Latin-Cyrillic Alphabet
0x00010006      ISO 8859-6:1987, Latin-Arabic Alphabet
0x00010007      ISO 8859-7:1987, Latin-Greek Alphabet
0x00010008      ISO 8859-8:1988, Latin-Hebrew Alphabet
0x00010009      ISO 8859-9:1989, Latin Alphabet No. 5
0x0001000a      /* reserved for future use */
0x0001000b      /* reserved for future use */
0x0001000c      /* reserved for future use */
[ . . .         . . .]
0x00010020      ISO 646:1991 IRV (International Reference
                Version)
[ . . .         . . .]
0x00010100      ISO 10646:1992, UCS-2, Level 1
0x00010101      ISO 10646:1992, UCS-2, Level 2
0x00010102      ISO 10646:1992, UCS-2, Level 3
0x00010104      ISO 10646:1992, UCS-4, Level 1
0x00010105      ISO 10646:1992, UCS-4, Level 2
0x00010106      ISO 10646:1992, UCS-4, Level 3
0x00010108      ISO 10646:1992, UTF-1, UCS Transformation
                Format 1
[ . . .         . . .]
0x00020001      OSF 10646:1992, UTF-2, File System Safe UCS
                Trans. Format
0x00020002      /* reserved for future use */
0x00020003      /* reserved for future use */
[. . .          . . .]
0x00020010      OSF UJIS (version to be defined)
0x00020011      OSF SJIS (version to be defined)
0x00020012      OSF Big 5 (version to be defined)
[ . . .         . . .]
0x00040001      JIS X0201:1976, Japanese phonetic characters
```

```
0x00040002      /* reserved for future use */
0x00040003      /* reserved for future use */
0x00040004      JIS X0208:1978, Japanese Kanji Graphic
                Characters
0x00040005      JIS X0208:1983, Japanese Kanji Graphic
                Characters
0x00040006      JIS X0208:1990, Japanese Kanji Graphic
                Characters
0x00040007      /* reserved for future use */
0x00040008      /* reserved for future use */
0x00040009      /* reserved for future use */
0x0004000a      JIS X0212:1990, Supplementary Japanese Kanji
                Graphic Chars
0x0004000b      /* reserved for future use */
0x0004000c      /* reserved for future use */
[. . .          . . .]
0x00040010      JIS AJEC:1991, Advanced Japanese EUC Code
[. . .          . . .]
0x00050001      KS C5601:1987, Korean Hangul and Hanja Graphic
                Characters
0x00050002      KS C5657:1991, Supplementary Korean Graphic
                Characters
0x00050003      /* reserved for future use */
0x00050004      /* reserved for future use */
[. . .          . . .]
0x0005000a      KS eucKR:1991, Korean EUC
[ . . .         . . .]
0x00060001      CNS 11643:1986, Taiwanese Hanzi Graphic
                Characters
0x00060002      CNS 11643:1992, Taiwanese Extended Hanzi
                Graphic Characters
0x00060003      /* reserved for future use */
0x00060004      /* reserved for future use */
[. . .          . . .]
0x0006000a      CNS eucTW:1991, Taiwanese EUC
0x00060010      CNS CSIC, Chinese Standard Interchange Code
```

3.5. Response to Registry Requests

OSF shall respond to all registry requests. Workload permitting, we expect to respond within four weeks with the newly registered value(s), a request for additional or clarifying information, or a rejection (with rationale). Reasons for rejection include, but are not limited to:

(a) Set already registered.

(b) Requested organization name already assigned.

(c) Registration request has insufficient information.

(d) Private company cannot register an international or national standard set.

(e) Bogus set.

An organization may appeal an OSF rejection. The rejection must include rationale or clarifying information that explains why the organization disagrees with OSF's decision. OSF will consider such requests, but it ultimately makes all final decisions regarding the registry.

3.6. Future Use and Vendor/User-Defined Values

Some ranges in the registry are reserved for unspecified future use by OSF, while others are designated for vendor/user-defined values. OSF reserves the organization values 0xf000 through 0xf4ff for possible future, unspecified enhancements to the registry.

The range 0xf500 through 0xffff is reserved for vendor- or user-defined values; OSF will never assign organization values in this range. This means the 32-bit values 0xf4ff0000 through 0xffffffff are available for organizations to identify sets that they don't want to register with OSF.

Values in the vendor/user-defined range are completely outside the control of OSF. There is no guarantee that the value a given organization selects has not already been selected by another organization. The way to guard against conflicting assignments is to register values with OSF. These values are only intended for use within a homogeneous environment.

4. MAINTENANCE, DISTRIBUTION, AND USE

This section covers maintenance and distribution mechanisms for the registry.

(a) *Version numbers.* The initial release of the registry is Version 1. Every time a new version is published, the version number is incremented.

(b) *Registry updates.* New versions are issued on an as-needed basis — that is, if there are sufficient additions to the list. Announcements of additions may be made between versions.

(c) *Upward compatibility.* Registry values never become obsolete. This means all registry versions are upward compatible because each contains all the information in all previous versions. For example, Version 5 would contain all entries in Versions 1 through 4, as well as whatever had been added for Version 5.

(d) *Registry distribution.* OSF maintains the file of registered values and distributes it on the source tape of each OSF technology in `/usr/lib/nls/loc/cs_registry`. In addition, the file contents are available from OSF on demand at no charge.

    The file contains selected information from the registry templates. The list of registered character sets appears first, followed by registered code sets.

(e) *Versions in OSF technologies.* In the future, OSF may require that implementations of OSF technologies support a certain version of the registry. Assume, for example, that Version 4 of the registry is in place before DCE R1.2 ships. OSF might require that an R1.2-based implementation use the Version 4 registered values for operations involving the exchange of code set identifiers. If Version 5 became available while licensees were shipping R1.2-based implementations, licensees could choose to support the newer version, but they would not be required to do so.

5. ADJUSTMENTS TO REGISTRY PROCEDURES

We have tried to determine what kind of information needed in character and code set registry requests, and the procedures to use in evaluating such requests. However, once actual requests begin coming in, it is possible we will discover that we need additional, or different, information. OSF reserves the right to adjust procedures as necessary to fix registry shortcomings. We will publicize any such changes.

6. CONVERTING TO AND FROM REGISTERED VALUES

Although the registry provides values that are consistent across a heterogeneous networks, individual operating systems will continue to use OS-specific strings to refer to code sets. Therefore, OSF provides two functions and specifies a source file syntax for mapping to and from registered values.

6.1. Mapping Fuctions

This section contains information about two new functions for mapping between local code set names and registered integer values. The functions are `dce_cs_loc_to_rgy()` and `dce_cs_rgy_to_loc()`.

(a) `dce_cs_loc_to_rgy()` — Convert local code set name to integer value.

```
void
dce_cs_loc_to_rgy(const char      *local_code_set_name,
                  unsigned32      *rgy_code_set_value,
                  error_status_t  *status);
```

This function accepts a string that holds the local name of a code set and returns the corresponding integer value of that code set, if the integer value exists. If no integer value exists for the supplied `local_code_set_name`, the value returned in `rgy_code_set_value` is undefined. The status parameter has one of the following values after the function returns:

(i) `dce_cs_c_ok` — The local code set name string is valid and an integer value was returned.

(ii) `dce_cs_c_unknown` — The local code set name string is unknown.

OSF defines and maintains most values of `rgy_code_set_value`, but implementations may add `rgy_code_set_value`'s in vendor- or user-definable ranges. In addition, all possible values of `local_code_set_name` are OS-specific and must be supplied when porting DCE to a given OS.

(b) `dce_cs_rgy_to_loc()` — Convert integer code set value to local string name.

```
void
dce_cs_rgy_to_loc(const unsigned32 rgy_code_set_value,
                  char             *local_code_set_name,
                  error_status_t   *status);
```

This function accepts an integer value of a code set, malloc's the appropriate space for the return `char *` value, and returns a NULL-terminated corresponding local code set name, if the local name exists. `local_code_set_name` is a maximum of 32 bytes — 31 character data bytes plus the terminating NULL. If no local name exists for the supplied `rgy_code_set_value`, the string returned in `local_code_set_name` is undefined. The status parameter has one of the following values after the function returns:

(i) `dce_cs_c_ok` — The local code set name string for the specified integer value was found and returned.

(ii) `dce_cs_c_unknown` — `rgy_code_set_value` is unknown

(iii) `dce_cs_c_notfound` — No local code set name exists for the supplied value of `rgy_code_set_value` (that is, the local code set name was the keyword NONE).

OSF defines and maintains most values of `rgy_code_set_value`, but implementations may add `rgy_code_set_value`'s in vendor- or user-definable ranges. In addition, all possible values of `local_code_set_name` are OS-specific and must be supplied when porting DCE to a given OS.

6.2. Mapping Table

The mapping functions described in the previous section use the information provided in an OS-specific version of a `code_set_table` file. This file contains the mapping of OSF-registered or user-defined code set values to the strings that a given OS uses when referring to those code sets. Table entries consist of three required fields:

(a) `loc_code_set_name`. A maximum 32-byte string (31 character data bytes plus a terminating NULL) that contains the OS-specific name of a code set or the keyword NONE. String values for OSF-registered code sets can contain characters from the PCS only; user-defined code set strings can contain any character.

(b) `rgy_code_set_value`. An unsigned32 that holds the registered or vendor/user-defined value for a code set. Vendor/user-defined values must be in the range 0xf4ff0000 through 0xffffffff.

(c) `char_set1[:...:char_setN]`. One or more unsigned16s that hold the registered or user-defined values of the character set(s) this code set encodes. Colons separate multiple values in this field.

One or more spaces or tabs separate fields within a table entry. Backslash (\) is the escape character and can be used to escape a newline to form line continuations. The values of `rgy_code_set_value` and `char_set` must be hexadecimal numbers only, but it is optional whether "0x" precedes each hex value.

Since the code and character set values are fairly cryptic, we recommend that a line of comments precede every table entry. The pound-sign (#) designates the beginning of comments.

Following is a brief excerpt of a `code_set_table`.

```
# ISO 8859-1:1987; Latin Alphabet No. 1
ISO8859-1   0x00010001   0x0011

# ISO 8859-2:1987; Latin Alphabet No. 2
ISO8859-2   0x00010002   0x0012

# ISO 10646:1992, UCS-2, Level 1
UCS2-L1     0x00010100   0x1000

# ISO 10646:1992, UCS-2, Level 2
UCS2-L2     0x00010101   0x1000

# AJEC:1991, Advanced Japanese EUC Code
AJEC        0x00020010   0x0011:0x0080:0x0081:0x0082

[. . .]

# User-defined code set; foo version of Japanese SJIS
fooSJIS     0xf0000001   0x0001:0x0080:0x0081
```

The local strings are more mnemonic and human-readable than are the registered integer values, but they differ from platform-to-platform, while the registered values remain consistent. For example, ISO 8859-1 has a single, OSF-registered value of 0x00010001, but it may have a name like one of these on a local system:

```
ISO8859-1
8859-1
iso88591
88591
Latin-1
```

This mapping table therefore allows each system to continue using its local names while ensuring there is a consistent way to refer to code sets in a distributed, heterogeneous network.

OSF supplies a partial version of `code_set_table` in `/usr/lib/nls/loc/code_set_table`. This partial version contains OSF-registered code set values, the character set(s) that the code sets encode, and comments. However, the local code set string value is the keyword NONE. Here is an excerpt from an OSF-supplied `code_set_table`:

```
# ISO 8859-1:1987; Latin Alphabet No. 1
NONE    0x00010001      0x0011

# ISO 8859-2:1987; Latin Alphabet No. 2
NONE    0x00010002      0x0012

# ISO 8859-3:1988; Latin Alphabet No. 3
NONE    0x00010003      0x0013

# ISO 8859-4:1988; Latin Alphabet No. 4
NONE    0x00010004      0x0014
```

```
# ISO 8859-5:1988; Latin Alphabet No. 5
NONE    0x00010005      0x0015

# ISO 8859-6:1987; Latin-Arabic Alphabet
NONE    0x00010006      0x0016

# ISO 8859-7:1987; Latin-Greek Alphabet
NONE    0x00010007      0x0017

# ISO 8859-8:1988; Latin-Hebrew Alphabet
NONE    0x00010008      0x0018

# ISO 8859-9:1989; Latin Alphabet No. 5
NONE    0x00010009      0x0019

# ISO 10646:1992, UCS-2, Level 1
NONE    0x00010100      0x1000

# ISO 10646:1992, UCS-2, Level 2
NONE    0x00010101      0x1000

[. . .]

# AJEC:1991, Advanced Japanese EUC Code
NONE    0x00020010      0x0011:0x0080:0x0081:0x0082

[. . .]

/* Registered values for these code sets are placeholders;
 * these organizations and sets have not yet been assigned
 * values. */

# pc850; IBM code page 850
NONE    0x01000001      0x0011

# ROMAN8; HP Western European code set
NONE    0x01010001      0x0011

[. . .]
```

When porting software to a given OS, vendors must replace the keyword NONE with the string name that each registered value has on the local platform. If no local name exists for a registered code set — that is, if the OS does not support the registered set — the keyword NONE must remain for that table entry.

In addition to replacing NONE with string values where appropriate, the vendor can add table entries for user-defined code sets.

The functions `dce_cs_loc_to_rgy()` and `dce_cs_rgy_to_loc()` use the first two fields in `code_set_table` entries. OSF technologies and application programs may also use the `char_set` values to evaluate whether code sets are compatible.

A completed `code_set_table` is stored at `/usr/lib/nls/loc/code_set_table`.

REFERENCES

[RFC 23] R. Mackey, DCE-RFC 23.0, ‘‘DCE 1.1 Internationalization Guide’’, January, 1993.

[RFC 27] S. Martin, DCE-RFC 27.0, ''Coded Character Set Conversions and Data Loss: Providing Interoperability While Preventing Loss'', December 1992.

AUTHOR'S ADDRESS

Sandra Martin
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
USA

Internet email: martin@osf.org
Telephone: +1-617-621-8707