TECHNICAL REPORT

ISO/IEC TR 15413

First edition
2001-03-15

Information technology — Font services — Abstract service definition

Technologies de l'information — Services de police de caractères — Définition du service abstrait

Reference number
ISO/IEC TR 15413:2001(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide by a simple majority vote of its participating members to publish a Technical Report. A Technical Report is entirely informative in nature and does not have to be reviewed until the data it provides are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this Technical Report may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 15413 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 34, *Document description and processing languages*.

Introduction

The technology to access large font libraries has become an increasingly important concern of document processing, electronic publishing, and application support facilities such as printing and display services.

This Technical Report specifies a system-independent interface to font services that provides for distribution, management and use of font resources.

This Technical Report is organized as follows:

— The general model of font services and its relationship to other parts of the document processing model are specified.

— The Abstract Font Service Interface (AFSI), which defines the facilities to access font resources in a system-independent way, is specified.

— Statements covering the conformance of font service systems are provided.

NOTE The first approach of JTC1/SC18 for Font Services was taken to develop an International Standard with two parts:

— Part 1: Abstract Service Definition

— Part 2: Protocol Specification.

During the standardization activities JTC1/SC18 found that the protocol for font resource interchange was still under technical development and changed the target to Technical Report for Font Services - Abstract Service Definition.

Information technology — Font services — Abstract service definition

1 Scope

This Technical Report provides the access facilities which can be used for creation, distribution, management, and use of font resources conforming to the architecture of ISO/IEC 9541.

This Technical Report is intended to be used in a variety of configurations meeting a variety of connectivity needs, including communication protocols, application programming interfaces, and application services.

This Technical Report defines an abstract interface to the font access facilities. This Technical Report will not specify the concrete syntax for a language binding of font service facilities, nor the concrete protocol used to communicate between the systems that provides or uses the font service facilities.

This Technical Report is intended for use in a wide variety of document processing environments, including:

— authoring;

— formatting and page layout;

— printing and display services;

— electronic publishing via removable media and/or information network.

2 References

ISO/IEC 9541-1:1991, *Information technology — Font information interchange — Part 1: Architecture*.

ISO/IEC 9541-3:1994, *Information technology — Font information interchange — Part 3: Glyph shape representation*.

ISO/IEC 10179:1996, *Information technology — Processing languages — Document Style Semantics and Specification Language (DSSSL)*.

ISO/IEC 10180:1995, *Information technology — Processing languages — Standard Page Description Language (SPDL)*.

3 Terms and definitions

For the purposes of this Technical Report, the following terms and definitions apply.

3.1
font manager
font service user that installs font resources into a system environment, manages the font usage and administers associated contracts (regarding access, royalties, etc.)

3.2
font property
property as defined in ISO/IEC 9541-1

3.3
font provider
font service user that designs typefaces, creates new font resources for sale and distribution (Design Source) or acquires font resources on consignment or for re-marketing with or without intervening modification (Data Source)

3.4
font query
query which is sent from a font service user to a font service entity to get or modify some font resource information

3.5
font requester
font service user that requests font resource information to create format, print, or display documents in a single or multi-system environment

3.6
font resource information
block of information about a font resource or some font resources which is returned from a font service entity to a font service user as the result of a font query

3.7
font service
interchange facility between a font service user and a server of font resources

3.8
font service API
interface between a font service user and a font service entity

3.9
font service entity
entity (or set of connected entities) which provides access facilities to resolve requests from font service users

3.10
font service protocol
relationship between font service entities to convey font service commands and responses

3.11
font substitution
substitution of an alternative font resource for a specified font resource which is not available

3.12
font service user
person and/or application software that provides requests, or manages font information in a computer system or network

3.13
glyph association
mapping or association of character codes (usually in an unformatted document data stream) to the glyph information contained in a font resource

3.14
glyph substitution
substitution (usually in a formatted document data stream) of an alternative glyph for a specified glyph which is not available in a requested font resource

4 Font service model

4.1 Font service user categories

Font service users are classified in the following three categories. In this Technical Report, the term "font service user" is used to represent all kinds of person(s) and application software in these categories.

Font Providers
Person(s) (and Application software) that design typefaces, create new font resources for sale and distribution (Design Source) or acquires font resources on consignment or for re-marketing with or without intervening modification (Data Source).

Font Managers
Person(s) (and Application software) that installs font resources into a system environment, manages the font usage and administers associated contracts (regarding access, royalties, etc.).

Font Requesters
Person(s) (and Application software) that requests font resource information to create format, print, or display documents in a single or multi-system environment.

4.2 Font service entity

Font service entity is an entity which provides facilities to access font resource information for font service users. This entity will be implemented as a program or component of a processing system which receives the font queries from font service users and returns font information.

A font service entity manages a set of font resources. Any information retrieval and/or modification on one of those font resources shall be done by this entity. In the case that a font service user sends a font query to a font service entity to get some information from a font resource which is not directly managed by the font service entity, the entity may send a font query to get the information from another font service entity. In such a case, a font service entity itself may act as a font service user to another font service entity.

Figure 4.1 shows a simplified model of the relationship between a font service entity and font service users.

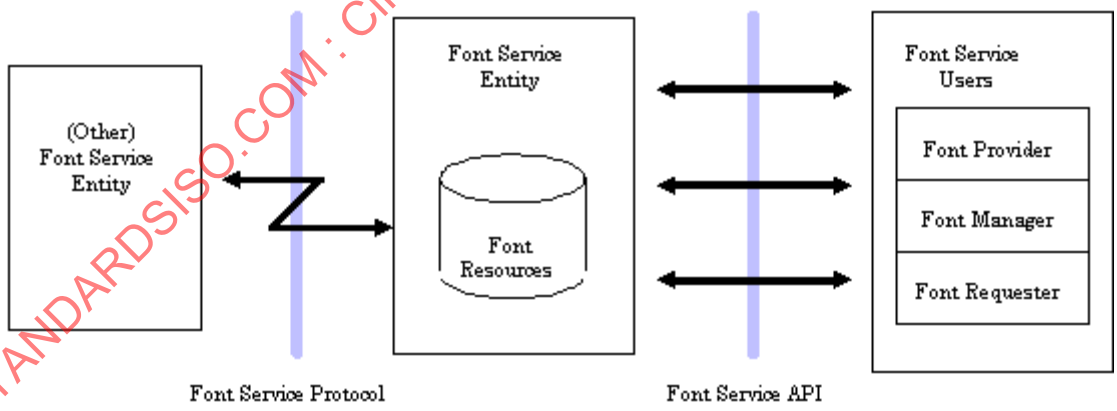

Figure 4.1 — Relationship between font service entity and font service users

Figure 4.2 shows the relationship between the font service entity and the ISO/IEC JTC 1/SC 34 document processing architecture.

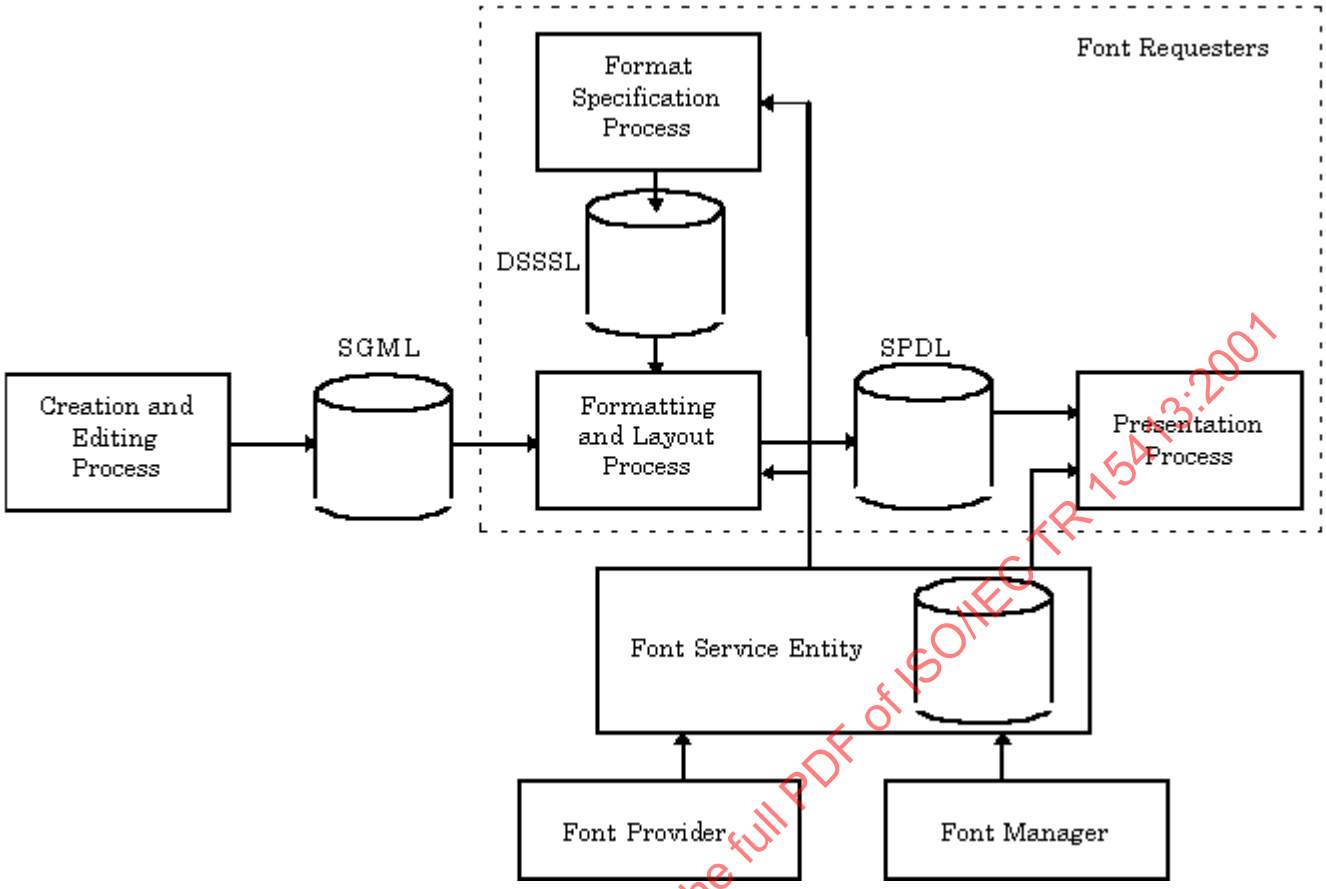

Figure 4.2 — Relationship between font service entity and SC 34 document processing architectures

The font resource information requested by these processes is different process by process. Formatting and Layout Process requests the description and metrics information. Presentation Process requests all the font resource information (description information, metrics information, and glyph shape information).

There may be a process called Font Installation and Maintenance Process, which provides new font resources to be managed by the font service entity. This process also executes the usage monitoring and the access control of each font resource.

4.3 Abstract Font Service Interface (AFSI)

This Technical Report defines the Abstract Font Service Interface (AFSI), which specifies the abstract interface between a font service user and a font service entity.

This Technical Report defines the AFSI as a set of specifications of "primitives". A primitive is a unit of communications between the font service entity and font service users. In this Technical Report, each primitive is defined as an abstract service which takes zero or more parameters, and returns zero or more results. Some primitives modify the internal data objects and/or the status of the font service entity as the side effect.

5 Data types

This clause defines data types used in the definitions of the Abstract Font Service Interface (AFSI).

Types used in AFSI are classified in two categories, base types and composite types. Base types are simple, elemental types and values of base types have no subordinate structures. Composite types are structures constructed from base types and/or other composite types.

5.1 Base types

5.1.1 Boolean

The set of values of type Boolean consists of two values, true and false. These values represent logical value true and false, respectively.

5.1.2 Integer

Each value of type Integer represents an integer number within the range -2^{32} to $2^{32} - 1$ inclusive.

5.1.3 Cardinal

Each value of type Cardinal represents an unsigned integer number within the range 0 to $2^{32} - 1$ inclusive.

5.1.4 Octet

Each value of type Octet represents an 8-bit byte data.

5.1.5 Handle

Each value of type Handle represents an unsigned integer number within the range 0 to $2^{32} - 1$ inclusive, which uniquely identifies a data object that exists in the font service entity. Font service users can access these data objects through Handles. The value 0 (zero) is reserved to indicate a NULL handle.

5.2 Composite types

5.2.1 OctetString

A value of type OctetString is an ordered sequence of octets.

5.2.2 Vector

A value of type Vector is an ordered sequence of values. Each value contained in a Vector may be a value of any type defined in this Technical Report.

5.2.3 PropertyList

A value of type PropertyList represents a corresponding value of the type property-list which is defined in ISO/IEC 9541-1.

5.2.4 UserInformationList

A value of type UserInformationList represents a corresponding value of the type user-information-list which is defined in detail, according to the following structure:

```
User Information List

ui_dataversion-userinformation ::= ui_dataversion-name, ui_dataversion-value
ui_datadversion-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//UI_DATAVERSION
ui_dataversion-value ::= CARDINAL

userid-userinformation-list ::= userid-name,userid-value-userinformatin-list
userid-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//USERID
```

```
userid-value-userinformation-list ::= ( username-userinformation, accesscode-use
rinformation, groupid-userinformation, usercomment-userinformation)*

username-userinformation ::= username-name,username-value
username-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//USERNAME
username-value ::= OCTET_STRING

accesscode-userinformation ::=accesscode-name,accesscode-value
accesscode-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//ACCESSCODE
accesscode-value ::= OCTET_STRING

groupid-userinformation ::=groupid-name,groupid-value
groupid-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//GROUPID
groupid-value ::= groupid-code
groupid-code ::= INTEGER { Font Providers  (0),
                           Font Managers   (1),
                           Font Requesters (2) }

usercomment-userinformation ::=usercomment-name,usercomment-value
usercomment-name ::= STRUCTURED-NAME
 -- ISO/IEC TR 15413//UESRCOMMENT
usercomment-value ::= OCTET_STRING
```

5.2.5 LicenseInformationList

A value of type LicenseInformationList represents a corresponding value of the type license-information-list which is defined in detail, according to the following structure:

```
License Information List

li_dataversion-licenseinformation ::= li_dataversion-name, li_dataversion-value
li_dataversion-name ::= STRUCTURED-NAME
-- ISO/IEC TR 15413//LI_DATAVERSION
li_dataversion-value ::= CARDINAL

expiredate-licenseinformation ::= expiredate-name, expiredate-value
expiredate-name ::= STRUCTURED-NAME
-- ISO/IEC TR 15413//EXPIREDATE
expiredate-value ::= (expireyear-licenseinformation, expiremonth-licenseinformation,
expireday-licenseinformation)

expireyear-licenseinformation ::= expireyear-name, expireyear-value
expireyear-name ::= STRUCTURED-NAME
-- ISO/IEC TR 15413//EXPIREYEAR
expireyear-value ::= OCTET_STRING

expiremonth-licenseinformation ::= expiremonth-name, expiremonth-value
expiremonth-name ::= STRUCTURED-NAME
-- ISO/IEC TR 15413//EXPIREMONTH
expiremonth-value ::= OCTET_STRING

expireday-licenseinformation ::= expireday-name, expireday-value
expireday-name ::= STRUCTURED-NAME
-- ISO/IEC TR 15413//EXPIREDAY
expireday-value ::= OCTET_STRING
```

6 Notation

This clause defines the notation used to define each primitive in clause 7.

For each primitive in this Technical Report, the primitive description specifies:

— the number and types of parameters required to invoke the primitive;

— the number and types of results;

— the relationship between the parameters and the results;

— any other effects the execution of the primitive has upon the internal data objects and/or the status of the font service entity.

7 Abstract Font Service Interface (AFSI)

7.1 Font service session control primitives

7.1.1 OpenSession

The **OpenSession** primitive takes one parameter:

user-id: OctetString;

and returns two results:

status: Cardinal;

session-id: Handle.

The **OpenSession** primitive establishes a session between a font service user and the font service entity for the purpose of accessing the font resource information.

The result status represents the completion status of the primitive. It takes one of the following values:

0 -- A session is opened successfully;

1 -- Failed to open a session;

10 -- Opening a session is denied.

The result session-id represents the handle which identifies the newly opened font service session. This result is void when the value of the result status is other than OK.

The value of session-id must be specified as one of the parameters for all other AFSI primitives.

7.1.2 CloseSession

The **CloseSession** primitive takes one parameter:

session-id: Handle;

and returns one result:

status: Cardinal.

The **CloseSession** primitive closes an opened session between a font service user and the font service entity.

The parameter session-id specifies an open session to be closed. The value of this parameter must be a value returned from the primitive **OpenSession** as the result session-id.

The result status represents the completion status of the primitive. It takes one of the following values:

0 -- The specified session is closed successfully;

2 -- The parameter session-id is invalid;

3 -- Failed to close the session (the parameter session-id is valid).

7.2 Font reference primitives

7.2.1 AssignFontReference

The **AssignFontReference** primitive takes two parameters:

session-id: Handle;

property-list: PropertyList;

and returns two results:

status: Cardinal;

font-reference: Handle.

The **AssignFontReference** primitive assigns the specified property-list to the specified font-reference. After the successful completion of this primitive, the assigned font-reference may be used as a parameter for query primitives instead of the property-list itself.

The result status represents the completion status of the primitive. It takes one of the following values:

0 -- The assignment is successful;

2 -- The parameter session-id is invalid.

7.2.2 ClearFontReference

The **ClearFontReference** primitive takes two parameters:

session-id: Handle;

font-reference: Handle;

and returns one result:

status: Cardinal.

The **ClearFontReferenceName** primitive cancels the assignment of the specified font-reference to a property list.

7.2.3 ListFontReferenceProperties

The **ListFontReferenceProperties** primitive takes two parameters:

session-id: Handle;

font-reference: Handle;

and returns two results:

status: Cardinal;

property-list: PropertyList.

The **ListFontReferenceProperties** primitive returns the property list which is associated with the specified font-reference by the **AssignFontReference** primitive.

7.2.4 ListAllFontReferences

The **ListAllFontReferences** primitive takes one parameter:

session-id: Handle;

and returns two results:

status: Cardinal;

font-reference-list: Vector of Handles.

The **ListAllFontReferences** primitive returns the list of all font references which are currently associated with property lists.

7.3 Availability check primitives

7.3.1 FindFontResource

The **FindFontResource** primitive takes two parameters:

session-id: Handle;

font-reference: Handle or PropertyList;

and returns two results:

status: Cardinal;

font-reference-list: Vector of Handles.

The **FindFontResource** primitive finds font resources which satisfy the conditions specified by the parameter font-reference, and returns the list of Handles of such font resources. The parameter font-reference may be specified as a Handle (font reference) or a PropertyList.

7.4 Font properties query primitives

7.4.1 GetFontProperty

The **GetFontProperty** primitive takes three parameters:

session-id: Handle;

font-resource: Handle;

property-list-specification: PropertyList;

and returns two results:

status: Cardinal;

required-property: PropertyList.

The **GetFontProperty** primitive retrieves the required properties from the font resource. The specification of the required properties is given by a PropertyList which contains property name and other conditions (if any). The list of all properties which satisfies the specification will be returned.

NOTE Since any property name defined in ISO/IEC 9541-1 can be specified in the parameter property-list-specification, this single primitive may be used to get any property contained in the font resource.

7.5 Font resource management primitives

7.5.1 PutFontProperty

The **PutFontProperty** primitive takes four parameters:

session-id: Handle;

font-resource: Handle;

property-list-specification: PropertyList;

new-property: PropertyList;

and returns one result:

status: Cardinal.

The **PutFontProperty** primitive replaces all of the specified properties in the font resource with the new property. The specification of the properties which are the subject to replacement is given by a PropertyList which contains the property name and other conditions (if any). All the properties which satisfy the specification will be replaced.

NOTE Since any property name defined in ISO/IEC 9541-1 can be specified in the parameter property-list-specification, this single primitive may be used to replace any property contained in the font resource.

7.5.2 CreateFontResource

The **CreateFontResource** primitive takes one parameter:

session-id: Handle;

and returns two results:

status: Cardinal;

new-font-resource: Handle.

The **CreateFontResource** primitive installs the new font resource into the font resource file with the font-property.

The result status represents the completion status of the primitive. It takes one of the following values:

0 -- The font resource is created successfully;

2 -- The parameter session-id is invalid.

7.6 Font service user management primitives

7.6.1 GetUserInformation

The **GetUserInformation** primitive takes three parameters:

session-id: Handle;

font-resource: Handle;

user-information-list-specification: UserInformationList;

and returns two results:

status: Cardinal;

required-user-information-list: UserInformationList.

The **GetUserInformation** primitive retrieves the required-user-information-list of the font-resource. The specification of the required user information is given by UserInformationList which contains the user names and other user information. The list of all user information which satisfy the specification will be returned.

7.6.2 GetLisenceInformation

The **GetLicenseInformation** primitive takes three parameters:

session-id: Handle;

font-resource: Handle;

license-information-list-specification: LicenseInformationList;

and returns two results:

status: Cardinal;

required-license-information-list: LicenseInformationList.

The **GetLicenseInformation** primitive retrieves the required-license-information-list of the font-resource. The specification of the required license information is given by LicenseInformationList which contains the license owner names and other license information. The list of all license information which satisfy the specification will be returned.

7.6.3 PutUserInformation

The **PutUserInformation** primitive takes four parameters:

session-id: Handle;

font-resource: Handle;

user-information-list-specification: UserInformationList;

new-user-information-list: UserInformationList;

and returns one result:

status: Cardinal.

The **PutUserInformation** primitive replaces all of the specified user-information-list in the font resource with the new user-information-list. The specification of the user-information-list which are the subject to replacement is given by user-information-list which contains the user name and other user information. All user-information which satisfies the specification will be replaced.

7.6.4 PutLicenseInformation

The **PutLicenseInformation** primitive takes four parameters:

session-id: Handle;

font-resource: Handle;

license-information-list-specification: LicenseInformationList;

new-license-information-list : LicenseInformationList;

and returns one result:

status: Cardinal.

The **PutLicenseInformation** primitive replaces all of the specified license-information-list in the font resource with the new license-information-list. The specification of the license-information-list which are the subject to replacement is given by license-information-list-specification which contains the license owner names and other license information. All license-information which satisfies the specification will be replaced.

Annex A

The abstract font service interface C language binding

A.1 Data types

A.1.1 Base types

a) AFS_Boolean

A AFS_Boolean is a 16-bit signed integer, either true (1) or false (0); all other values are undefined.

C Binding

typedef short AFS_Boolean;

b) AFS_Int32

A AFS_Int32 is a 32-bit signed integer within the range -2^{31} to $2^{31} - 1$.

C Binding

typedef long AFS_Int32;

c) AFS_Card32

A AFS_Card32 is a 32-bit unsigned integer within the range 0 to $2^{32} - 1$.

C Binding

typedef unsigned AFS_Card32;

d) AFS_Octet

A AFS_Octet is a 8-bit value.

C Binding

typedef char AFS_Octet;

e) AFS_Handle

A AFS_Handle is a 32-bit unsigned integer within the range 0 to $2^{32} - 1$ which uniquely identifies for a private data structure managed by the Font Service Entity.

C Binding

typedef AFS_Card32 AFS_Handle;

A.1.2 Composite types

a) AFS_OctetString

AFS_OctetString is a fixed length string consists of an ordered sequence of AFS_Octet[s].

C Binding

typedef AFS_Octet AFS_OctetString[AFS_MAX_STRING];

b) AFS_Property

AFS_Property is a data structure for a property defined in ISO/IEC 9541-1. A property represents font resource information. member propName in AFS_Property stores the unambiguous property name. propType stores the data type of the property value. When the data type is numerical data (except binary stream), the property value is stored in propValue1 and propValue2. In case of string or binary stream data, it is stored in other area, and propValue1 stores offset value to the area that substantial data is stored, and propValue2 stores data length in byte.

C Binding

typedef struct {
AFS_OctetString propName;
AFS_Card32 propType;
AFS_Card32 propValue1;
AFS_Card32 propValue2;
} AFS_Property;

c) AFS_PropertyList

AFS_PropertyList is a list of zero or more related AFS_Property[s], the value of some other higher-level property, which may be specified to be ordered in a particular sequence.

C Binding

typedef AFS_Property *AFS_PropertyList;

d) AFS_UserInformationList

AFS_UserInformationList is a data structure that represents User-Information-list.

e) AFS_LicenseInformationList

AFS_UserInformationList is a data structure that represents License-Information-list.

A.2 Application program interface

A.2.1 Font service session control functions

a) OpenSession

Syntax

AFS_Card32 AFSOpenSession(userID, pSessionID);

AFS_OctetString userID;

AFS_Handle *pSessionID;

Arguments

userID - User ID.

Access:read-only

pSessoinID - The Handle that uniquely identifies the session between a Font Service User and the Font Service Entity.

Access:write-only

Return Values

AFS_SUCCESS -- A session is opened successfully;

AFS_ACCESS_DENIED -- Access denied;

AFS_FAILURE -- Failed to open a session because of any reason.

Description

Establishes a session between a Font Service User and the Font Service Entity for accessing the font resource information. This function stores the unique handle in the address pointed by argument pSessionID.

This function should be called first before other AFSI functions called.

b) CloseSession

Syntax

AFS_Card32 AFSCloseSession(sessionID);

AFS_Handle sessionID;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

Return Values

AFS_SUCCESS -- The specified session is closed successfully;

AFS_SESSION_INVALID -- Argument session ID is invalid;

AFS_CLOSE_FAILED -- Failed to close the session (the argument session ID is valid).

Description

Closes a session between a Font Service User and the Font Service Entity.

A.2.2 Font reference functions

a) AssignFontReference

Syntax

AFS_Card32 AFSAssignFontReference(sessionID, propList, propString, pFontRef);

AFS_Handle sessionID;

AFS_PropertyList propList;

AFS_Octet *propString;

AFS_Handle *pFontRef;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

propList - A property-list.

Access:read-only

propString - The property value stores string or binary stream data.

Access:read-only

pFontRef - A Pointer to the unique identifier specifies assigned property-list.

Access:write-only

Return Values

AFS_SUCCESS -- Successful;

AFS_SESSION_INVALID -- Argument session ID is invalid;

AFS_MEMORY_ERROR -- Memory error occurred.

Description

Assigns the specified property-list to a Handle called font-reference. After the successful completion of this function, the assigned font-reference may be used as a parameter for query functions instead of the property-list itself.

b) ClearFontReference

Syntax

AFS_Card32 AFSClearFontReference(AFS_Handle sessionID, AFS_Handle fontRef);

AFS_Handle sessionID;

AFS_Handle fontRef;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

fontRef - The Handle identifies the specified property-list.

Access:read-only

Return Values

AFS_SUCCESS -- Successful;

AFS_SESSION_INVALID-- Argument session ID is invalid;

AFS_FONT_REFERENCE_INVALID-- Argument font-reference is invalid.

Description

Cancels the assignment of the specified font-reference to the property-list.

c) ListFontReferenceProperites

Syntax

AFS_Card32 AFSListFontReferenceProperties(sessionID, fontRef, pPropList, propString);

AFS_Handle sessionID;

AFS_Handle fontRef;

AFS_PropertyList *pPropList;

AFS_Octet *propString;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

fontRef - The Handle identifies the specified property-list.

Access:read-only

pPropList - The property-list that is associated with fontRef.

Access:write-only

propString - The property value stores string or binary stream data.

Access:write-only

Return Values

AFS_SUCCESS -- Successful;

AFS_SESSION_INVALID-- Argument session ID is invalid;

AFS_FONT_REFERENCE_INVALID -- Argument font-reference is invalid;

AFS_MEMORY_ERROR -- Memory error occurred.

Description

Returns the property-list that is associated with the specified font-reference.

d) ListAllFontReferences

Syntax

AFS_Card32 AFSListAllFontReferences(sessionID, pFonRefVec);

AFS_Handle sessionID;

AFS_Handle *pFontRefVec;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

pFontRefVec - A Pointer to an array that stores font-reference[s].

Access:write-only

Return Values

AFS_SUCCESS -- Successful;

AFS_SESSION_INVALID-- Argument session ID is invalid;

AFS_MEMORY_ERROR -- Memory error occurred.

Description

Returns the list of all font-reference[s] assigned in the session.

A.2.3 Availability check functions

a) FindFontResource

Syntax

AFS_Card32 AFSFindFontResource(sessionID, fontRef, pFontResVec);

AFS_Handle sessionID;

AFS_Handle fontRef;

AFS_Handle *pFontResVec;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

fontRef - The Handle identifies the specified property-list for finding font resources.

Access:read-only

pFontResVec - A Pointer to an array that stores found font resources.

Return Values

AFS_SUCCESS -- Successful;

AFS_SESSION_INVALID-- Argument session ID is invalid;

AFS_FONT_REFERENCE_INVALID -- Argument font-reference is invalid;

AFS_MEMORY_ERROR -- Memory error occurred.

Description

Finds font resources match to the font-property specified the font-reference, and returns the list of such font resources.

A.2.4 Font properties query functions

a) GetFontProperty

Syntax

AFS_Card32 AFSGetFontProperty(sessionID, fontRes, propList, propString);

AFS_Handle sessionID;

AFS_Handle fontRes;

AFS_PropertyList propList;

AFS_Octet *propString;

Arguments

sessionID - The Handle that identifies for a session that AFSOpenSession() returns.

Access:read-only

fontRes - The Handle that uniquely identifies the specifies font resources.

Access:read-only

propList - The property-list specifies properties that retrieve from the Font Service Entity. And stores properties that satisfies the specification as output.

Access:read and write