DSM-CC Client Programmer's Reference Guide

11 November 1997

Bionic Buffalo Corporation

2533 North Carson Street, Suite 1884
Carson City, Nevada 89706-0147 USA

http://www.tatanka.com

Document Revision History

Original release, 7 August 1997.

Update, 2 September 1997.

Update, 13 October 1997.

Update, 2 November 1997. Notes on compliance.

Update, 11 November 1997. Expanded explanations of API. Added Chapter, "Client Applications".

Printed 14:50:48, Thursday, 14 May, 1998.

Additional Document Information

Project name: *spain*.

Document file name: *esref02.doc*.

Written with Microsoft Word 6. Illustrations created with Visio Technical 4.5. Set in Monotype Bulmer, Adobe Ocean Sans, and Adobe Tekton.

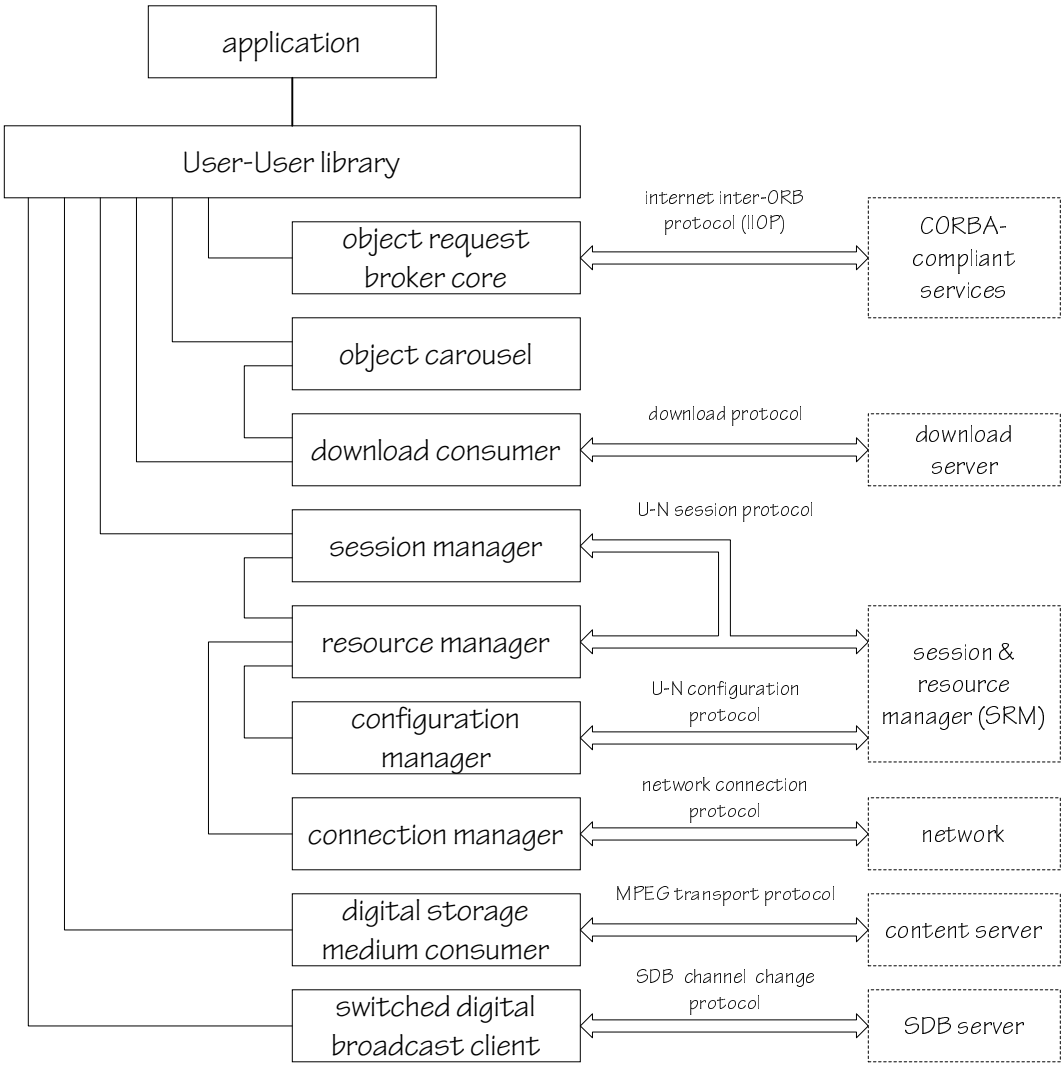
Contents

Overview

This document describes the client implementation of DSM-CC, which is described by ISO/IEC 13818-6. This implementation corresponds to the recommendations of DAVIC 1.2.

The client API is also available to server applications, and in fact constitutes the majority of the server API as well. Therefore, this document is also a reference for the DSM-CC server.

The basic construction of a DSM-CC client is shown here:



The only application programming interface (API) defined by the standards for applications is the User-User API. The other components APIs are proprietary, except for a few procedures which may not be present in all implementations. By using only the User-User API, a client application is portable among different platforms.

Likewise, not every environment or implementation includes access to all protocols or components. However, the User-User API provides well-defined mechanisms that allow an application to determine if a particular service is present and available.

DSM-CC is not a single protocol. ISO/IEC 13818-6 specifies a number of different protocols, including download, user-user, and others. Their primary functions are:

- Internet Inter-ORB Protocol (IIOP) is the primary protocol for access to high-level application services hosted on remote platforms. IIOP is platform-independent, so the application does not necessarily connect to a specific hardware server.

- User-Network Configuration allows an environment to set configuration parameters for a client in a proprietary fashion. The use of this protocol is not portable.

- User-Network Session manages sessions between clients and servers. A session includes any relationship between the client software and the server object. A server object is a software entity, not necessarily associated with a single hardware platform. Therefore, each use of a remote object potentially requires a separate session.

- Download provides transfer of information from servers to clients over a (possibly) asymmetric channel. Typically, objects are downloaded over the high-speed channel used for content, in cases where the bi-directional channel used for IIOP does not have the speed required for adequate performance. The download protocol has two parts: download data (typically high bandwidth) and download control (requiring lower bandwidth).

- The network connection protocol is defined by the underlying network, and is referenced by the DSM-CC specification but not defined by it. An example is the Q.2931 protocol used for ATM signalling.

- MPEG transport protocol, defined by ISO/IEC 13818-1, is used to carry audio, video, data, and other content to the client.

- Switched-Digital Broadcast (SDB) channel change protocol is used in some environments where a client must request action from an Interworking Unit (IWU) to effect channel connection changes.

DAVIC allows various protocol stack configurations for the realization of a network. Information flows are classified into groups:

- S1 (content) flow: MPEG audio and video, download data

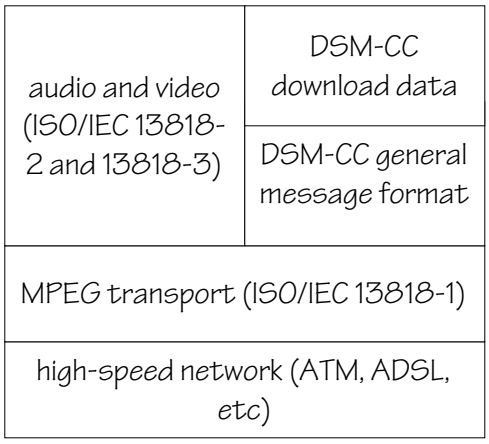
- S2 (application control) flow: IIOP (CORBA-compliant services, application-level authentication), download control

- S3 (session control) flow: user-network (including session-level authentication)

- S4 (network) flow: signalling (Q.2931)

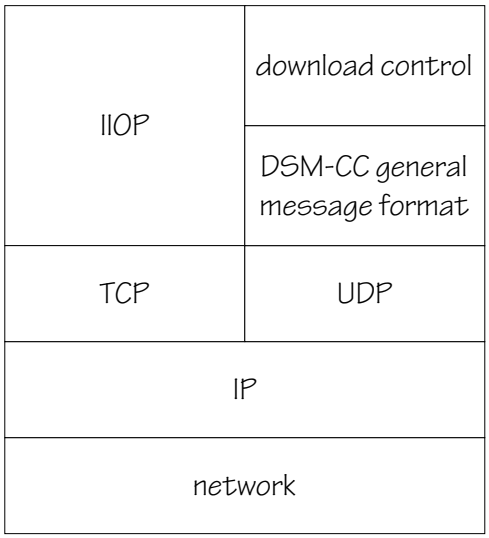
Although there are four flows, they may be combined for transmission over only one or two channels. For example, it is common to combine S2, S3, and S4 on a lower speed bi-directional channel, while placing the S1 flow on a high-speed, uni-directional channel. (Note that the download protocol is often split between two different channels, since it uses both S1 and S2 flows.

The most common protocol stack configuration for the S1 (content) flow is:

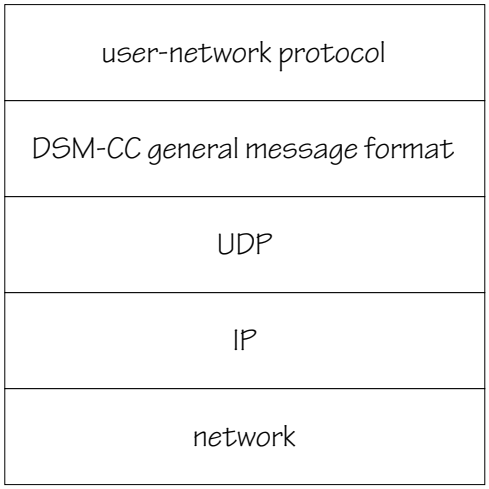


The DSM-CC general message format refers to an overall structure for DSM-CC messages. In the S1 flow, they are encapsulated within the private data section of the MPEG transport stream.

The S2 flow requires a reliable transport mechanism, which is usually TCP/IP. The typical stack for the S2 flow is:



The S3 flow generally follows the same approach as is used for download control:



Client Applications

Using CORBA

CORBA (Common Object Request Broker Architecture) is the foundation for the DSM-CC API. CORBA provides a uniform mechanism for defining and using objects.

Object interfaces are described using IDL (Interface Definition Language). An object description can be mapped from IDL to common programming languages such as C, C++, COBOL, Ada, Java, and Smalltalk. A client application written in any of these languages can use objects written in any other of the languages. The Bionic Buffalo implementation of DSM-CC uses the C-language mapping of the DSM-CC IDL

Every object has an *interface*, an *implementation*, and an *identity*.

Interface

IDL is used to define the interface of members of an object class. The interface of an object consists of the externally-visible attributes and operations of that object, and the data structures associated with them. The programming interface to any object with a given interface is the same as that for any other object with the same interface.

In C, each operation of an object maps to a procedure. Each attribute of an object maps to two procedures: one to get the value of the attribute, and one to set the value of the attribute. (*Readonly* attributes do not have "set" routines.)

Implementation

The implementation of an object is the program which performs the operations defined on the object's interface. Different objects may share the same implementation, and the implementation of an object can be changed during the object's lifetime.

Identity

The identity of an object specifies which object is considered. An object may have an identity which is independent of any particular implementation, program, server, or geographic location.

In the C language, an opaque data type ***CORBA_Object*** is used to refer to a specific object.

(Note the CORBA object model is different from some other object models such as Microsoft's Common Object Model, or COM. COM does not maintain the distinction between identity and implementation. By some definitions, Microsoft objects are not "true" objects. However, it is possible to write CORBA programs which access Microsoft objects through a bridge.)

It is important that CORBA objects have no specific location or implementation. When an application has an object reference (of type ***CORBA_Object***), the application has no general mechanism to discover if the object is local or remote, or if the implementation is on one machine or on many.

The first argument to every C procedure defined by DSM-CC is a ***CORBA_Object*** value. This specifies *which* object is referenced by the procedure.

CORBA defines a data structure ***CORBA_Environment*** for the C mapping. The ***CORBA_Environment*** is an opaque structure, except for the first element:

```
typedef struct CORBA_Environment
        {
        CORBA_exception_type        _major ;
        ... /* implementation-specific */
        }
        CORBA_Environment ;
```

A pointer to the ***CORBA_Environment*** structure is passed as the last parameter to all C procedures defined by DSM-CC. The main use of the ***CORBA_Environment*** structure is to handle *exceptions*, which are described below.

The user is referred to the Bionic Buffalo *CORBA API* reference manual for additional information.

Exceptions

After invoking an operation on a CORBA object, an application checks the value of the ***_major*** element of the ***CORBA_Environment*** structure. The ***_major*** element can have one of three possible values:

CORBA_NO_EXCEPTION indicates the operation performed successfully

CORBA_SYSTEM_EXCEPTION indicates the operation encountered one of the standard system exceptions defined by CORBA

CORBA_USER_EXCEPTION indicates the operation encountered an application-specific exception (in this case, defined by DSM-CC)

Any of the standard system exceptions can occur for any operation. The specific user exceptions which can occur for an operation are defined by the operation's IDL.

If the application discovers that an exception has occurred, then the application can invoke ***CORBA_exception_id()*** to learn which exception. Each exception is identified by its *repository identifier*, which is a unique string name for the exception. The programmer should use the exception name constants ***#define***d in the DSM-CC and CORBA header files.

Some exceptions are associated with data structures. An application can retrieve the data structure associated with an exception by calling ***CORBA_exception_value()***. Exception data is used to pass additional information about the exception to the application.

Naming Objects

CORBA objects by themselves do not have names. However, a separate class of objects based on the CORBA *Name Service* can be used to associate names with objects externally.

Name Service objects are roughly akin to the directories of traditional file systems. The names of files are not kept in the files themselves, but rather in directories. Similarly, the names of objects are not kept with the objects themselves, but rather in Name Service objects. In DSM-CC, ***DSM_Directory*** objects and ***DSM_ServiceGateway*** objects inherit the same functions, with a few additional operations.

Neither DSM-CC nor DAVIC standardize the ways in which directories on a server must be organized. Finding the desired file, stream, or other objects in a Service Gateway or Directory is an implementation-specific process.

Initializing the Environment and the ORB

The first actions required of a DSM-CC application are to initialize the ***CORBA_Environment*** and the ***CORBA_ORB***.

CORBA does not standardize the mechanism for initializing the CORBA environment. The implementation of CORBA used in Bionic Buffalo's DSM-CC uses the routine ***eg_new_environment()*** to allocate and to initialize the ***CORBA_Environment*** data structure.

Once the environment is established, the next step is to initialize the ORB. CORBA defines the routine ***CORBA_ORB_init()*** for this purpose. This routine returns the ***CORBA_Object*** reference value for the ORB itself.

Finding the First Objects

The application can acquire an initial list of services using the procedure ***CORBA_ORB_list_initial_services()***. These services are represented as strings.

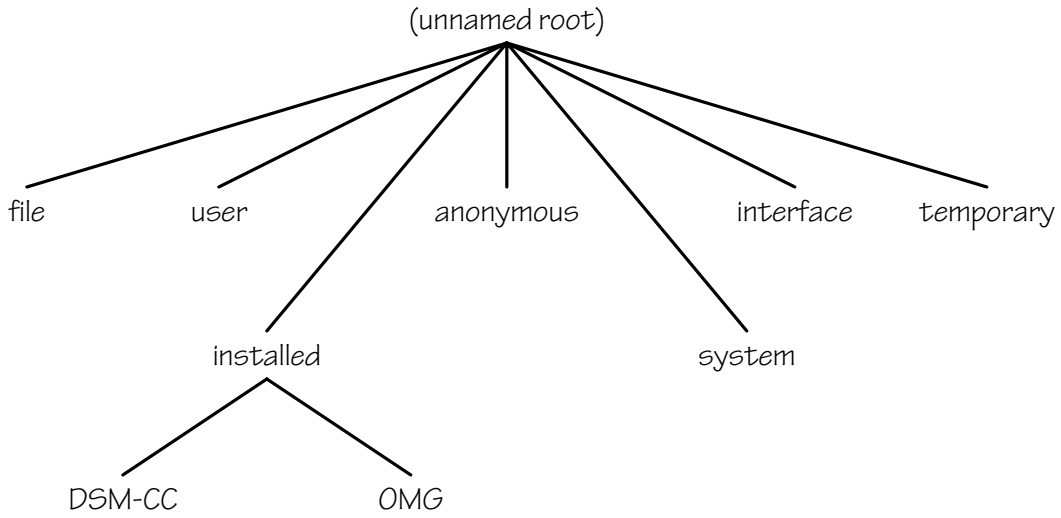
The first object needed by a DSM-CC application is a ***DSM::Session*** object. To find the Session object, the application must use the CORBA Name Service. This procedure is similar to looking up the Session object in a directory. The Name Service object, which is equivalent to the root directory, is represented by the string "NameService" in the list returned by the procedure ***CORBA_ORB_list_initial_services()***.

The procedure ***CORBA_ORB_resolve_initial_references()*** will convert the string name of the Name Service object to a ***CORBA_Object*** reference. For this example, we will assign this value to the variable ***root_directory***, which we declare as

```
CosNaming_NamingContext          root_directory ;
```

(The Name Service uses a Naming Context object in the same way a directory is used by a file system.) Once the application has the ***root_directory*** object, it can use the ***CosNaming_NamingContext_list()*** operation to discover the objects in the root directory.

Just as in a file directory system, Naming Context objects can be nested. The standard Bionic Buffalo object directory tree has the following partial structure:



Each node of the tree in the above diagram represents a Naming Context object. Each Naming Context object is given a name by the Naming Context object above it in the tree. The root Naming Context object is given no name.

The needed Session object will be in the DSM-CC Naming Context, and will have the name Session.

To traverse the tree, the application must iteratively invoke the ***CosNaming_NamingContext_list()*** operation on Naming Contexts in the path. At each step, passing the Context's name to ***CosNaming_NamingContext_resolve()*** will yield the object reference for the next step.

Other Naming Contexts which might be useful are:

- the ***file*** context, which (if the appropriate software is installed) will contain an object-oriented version of the local file system directory tree
- the ***OMG*** context, which contains various CORBA services
- the ***interface*** context, which is an alternative view of the interface repository

It should be noted that security features and implementation decisions may restrict the visibility of objects in any Naming Context.

Establishing a Session

Once the ***DSM_Session*** object has been located, the ***DSM_Session_attach()*** operation can be called to establish a session with a server. This operation returns a sequence of object references, one of which will be to a ***DSM_First*** object. The ***DSM_First*** object will have two operations (***root*** and ***service***) which will return the Service Gateway and primary service objects for the current session.

The application's behaviour at this point is dependent upon the implementations of the client and the server. The specification does not require any specific initial application or first service, nor does it specify a default operation of the first service to launch any initial application.

Carousels and Downloaded Objects

An application normally does not distinguish among the various "sources" of objects. Whether an object is defined locally or remotely, and whether it is downloaded or found in an object carousel is typically transparent to the application. No special programming is required, and the User-User API does not differ from one object implementation to the next.

Notes on Implementation and Use

Extended Profiles in the User-User Protocol

A DSM-CC client application views the external world as a collection of objects. In general, it does not know the location or other details of implementation of these objects. However, the other components of the client (the resource manager, video decoder, and so on) must be aware of the specifics of the implementation.

The association between objects (at the application level) and resources (in the other client DSM-CC components) is based on an extension to the CORBA Internet Inter-ORB Protocol (IIOP). The extension is transparent to CORBA applications, but embeds extra information into the IIOP messages so that other client components can extract the association between objects and resources.

At the protocol level, object references are encoded as Interoperable Object References (IORs). To an application, an object reference (in the C language mapping) is encoded as a pointer to void (***void*** *). The application pointer does *not*, in general, reference an object's IOR. One of the functions of the ORB is to translate between the internal representation of objects (which is defined by the ORB's implementor) and the external representation of objects (which is the IOR).

An IOR is encoded as a type identifier plus a series of zero or more tagged profiles. The type identifier is a string, such as "DSM::Stream". If the reference is to a null object, the string is empty, and there are no tagged profiles. Otherwise, there is at least one tagged profile.

The first field of each tagged profile is a tag, which identifier the type of profile. Tag numbers are assigned by the Object Management Group (OMG), the same body which defines the CORBA standard. Although multiple tagged profiles might be present, they all refer to the same object. Each profile represents a different approach to the object.

An internet profile, with an assigned tag number of zero, contains an internet address or host name, a port number, and an opaque key. Together, these point to the location of the object. (The opaque key is defined on the system where the object was created, and has no significance elsewhere.) Two internet profiles might exist in a single IOR when the same object can be accessed at two different internet addresses.

DSM-CC has registered a number of profile tags for different purposes, although only some are actually used in a DAVIC-compliant implementation. These profiles greatly extend the domain of an IOR as used in DSM-CC. Some of the additional information which can be carried includes:

- ***DSM::ConnBinder***, which is a sequence of ***DSM::Tap***. This structure includes fields which correspond to the resource identifiers used in the User-Network protocol.

- ***BIOP::ObjectLocation***, which locates an object within a broadcast network. This is used with object carousels.

- ***DSM::IntfCode***, which describes interfaces in a repository.

Most of this information is never needed directly by an application. Some of it is supplied to applications by way of the User-User API. However, the resource information from the User-Network protocol is necessary for implementation of critical components such as the video decoder and the interface to the network. For these purposes, a proprietary API has been provided.

Each resource corresponds to an object. A list of the resource objects associated with a User-User object can be obtained through the routine ***es_rsrc_list()***. Each of these resource objects may, in turn, be interrogated for details.

For example, if a ***DSM::Stream*** object is passed to ***es_rsrc_list()***, one of the returned resource objects will should be an MPEG program resource object. (Assuming that the server and SRM implementations of the User-Network protocol have passed the appropriate resource descriptor to the client.) Using ***es_rsrc_mpeg_pgm_info()***, an video decoder can recover the PID and other information for the ***DSM::Stream*** object.

The resource objects may not always be available, since they are not all mandatory in all implementations. Some environments rely on external mechanisms for conveying the resource information, and some platforms make assumptions about the configuration based on predefined parameters.

Service Context

In each User-User request and reply message, there is a Service Context List. This is a sequence of Service Contexts. This list is not directly exposed to the application program.

DSM-CC defines five types of Service Context:

- *compatibility descriptor*, for operation-specific compatibility information exchange

- *download information request*, to negotiate download protocol parameters in a request

- *download information response*, to negotiate download parameters in a reply

- *authorization request*, used in security and authentication procedures

- *connection binder*, used to describe download resources

- *version information*, for use in a resolve request, to select among various versions when the latest version is not requested

The ***es_service_context_set()*** and ***es_service_context_get()*** procedures are used to access these structures when necessary.

Download

DSM-CC defines a download protocol, and also a ***DSM::Download*** interface. The interface is an explicit request for use of the protocol, but download operations may also occur implicitly under various circumstances.

Explicit Request

A client may explicitly request a download operation using the ***DSM::Download*** interface. This interface may be used with objects of class ***DSM::Download***, and will cause the downloaded object to be placed into memory buffers allocated for that purpose.

The ***DSM::Directory*** operations may be used to determine which objects have the ***DSM::Download*** interface.

Implicit Request

A transfer request such as a ***DSM_File_Read()*** may be satisified using the download protocol instead of by means of IIOP. This is useful when the download channel (S1 flow) is significantly faster than the IIOP channel (S2 flow), and the amount of data to be transferred is large.

To effect this, the client places download request information in the service context list of the transfer request. The server response may include download response information in the reply message to allow the download operation.

This procedure is initiated by the User-User library when appropriate, as defined by configuration parameters.

Beginning of Session

When a session is initiated, the User-Network ***ClientSessionSetupRequest*** message contains proposed download protocol parameters. The returned ***ClientSessionSetupResponse*** message contains a series of references to objects to be downloaded, along with the download connection information and the negotiated download parameters. The User-User library downloads these objects before returning to the application.

This procedure occurs without intervention of the client application.

Security and Authentication

Most DSM-CC object inherit ***DSM::Access***, which includes a structure attribute of type ***Perms_T***. This structure has ***OWNER*** access, so it is not generally visible to the client. Two fields relate to authentication:

- ***aPassword*** is a character string required for access to the object
- ***authData*** is opaque data used in an implementation-dependent fashion for an encrypted challenge sequence to the client when the object is accessed

The authentication requirements can apply to any object, whether it be a file, stream, or an entire server via a service gateway.

When such an object (with a non-null password or non-empty ***authData***) is referenced, an authentication request is passed within the service context list of the resolve operation. Attempts to use the object will result in a ***DSM::NO_AUTH*** exception. The client is assumed to understand the reason for the exception. After invoking ***DSM_Security_authenticate()*** to respond to the authentication request, the failed request may be resubmitted by the client.

Compliance with Applicable Standards

Baseline Documents

Bionic Buffalo DSM-CC is coded according to the following standards:

ISO/IEC 13818-6, Digital Storage Media Command & Control (DSM-CC), 12 July 1996.

> This document is the basic definition of DSM-CC.

OMG 97-09-01, Common Object Request Broker: Architecture and Specification, Revision 2.1, September 1997.

> This defines CORBA, the basis for the DSM-CC User-User protocol.

DAVIC, DAVIC 1.2 Specification, 1997.

> This defines the overall architecture of the system, including services and content provision, transmission, and consumption.

There are numerous other specifications referenced by these documents, but these are the most significant.

Selection of Options

All of the standards documents mentioned in the previous section allow a choice of options, features, and implementation mechanisms. The DAVIC 1.2 Specification narrows the range of choices to define an overall architecture which provides high levels of interoperability.

As much as practical, the DAVIC 1.2 Specification has been used to select from among the options and choices.

Resolution of Conflicts and Ambiguities

The most significant of the conflicts among the specifications are, in fact, minor. They result from inconsistencies between the CORBA 2.1 specification and the DSM-CC specification. This implementation resolves them in favor of the CORBA approach because:

1. The DSM-CC specification frequently takes the position that the CORBA approach controls matters involving CORBA; and

2. Compliance with CORBA allows for interoperability with other CORBA systems.

These discrepancies do not affect interoperability with other DSM-CC systems: they relate only to the DSM-CC Application Programming Interface (API).

C-Language Mapping of DSM-CC IDL

Early versions of CORBA mapped IDL operations to C procedures so that the environment variable was always the second argument. For example, the IDL

```
module DSM
        {
        interface Stream
                {
                void play ( in AppNPT rStart, in AppNPT rStop, in Scale rScale ) ;
                }
        }
```

mapped to the C procedure

```
DSM_Stream_play ( DSM_Stream stream_object,
                        CORBA_Environment * env1,
                        DSM_AppNPT * rStart, DSM_AppNPT * rStop,
                        DSM_Scale * rScale ) ;
```

This is the same approach taken in the DSM-CC specification. However, the newer CORBA specifications map the environment variable to the final argument:

```
DSM_Stream_play ( DSM_Stream stream_object,
                        DSM_AppNPT * rStart, DSM_AppNPT * rStop,
                        DSM_Scale * rScale,
                        CORBA_Environment * env1 ) ;
```

The newer approach is adopted by this implementation. This is compatible with CORBA-compliant programming environments since CORBA 2.0 (July 1995).

Note that this does not affect interoperability with other DSM-CC implementations, since it does not alter the protocol.

Output of the *DSM::LifeCycle::create* Operation

The IDL for the object creation operation in DSM-CC is:

```
module DSM
        {
        interface LifeCycle
                {
                void create ( in string aKind, in Version rVersion, out IOP::IOR rObjRef ) ;
                }
        }
```

The commentary explains that the resulting IOR can be used (for example) with ***DSM::Directory::bind***. However, this is inconsistent both with the CORBA and CORBA services specifications, and with the DSM-CC specification itself.

1. Examination of the ***DSM::Directory::bind*** documentation shows that the input is not an IOR, but an item of type ***DSM_ObjRef***. The DSM-CC specification makes it clear that the CORBA services specification controls.

2. The CORBA specification, which defines the format of an IOR, explains that IORs normally should be opaque to application programmers. This is contrary to the explicit use in the DSM::LifeCycle::create definition.

3. The CORBA services Life Cycle service, while not directly the basis for the DSM interface, produces standard opaque object references, *not* IORs.

4. No other DSM-CC application function refers to an IOR, which makes the result of ***DSM::LifeCycle::create*** (as defined by the specification) useless for most purposes.

For these reasons, this implementation uses a slightly different definition of the ***create*** operation:

```
module DSM
        {
        interface LifeCycle
                {
                void create ( in string aKind, in Version rVersion, out Object rObjRef ) ;
                }
        }
```

In this way, ***rObjRef*** is compatible with other DSM-CC operations and data structures.

Note that this does not affect interoperability with other DSM-CC implementations, since it does not alter the protocol.

Organization of Software Components

(to be supplied)

Application Programmer Interface (API)

CosNaming - data structures for the naming service

Synopsis - C

```
#include        <dsmcc.h>

typedef char                            * Istring ;         /* identifier string */

typedef struct
        {
        Istring                 id ;
        Istring                 kind ;
        }
        CosNaming_NameComponent ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CosNaming_NameComponent         * _buffer ;
        }
        CosNaming_Name ;

typedef unsigned long                   CosNaming_BindingType ;

typedef struct
        {
        CosNaming_Name                  binding_name ;
        CosNaming_BindingType           binding_type ;
        }
        CosNaming_Binding ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CosNaming_Binding               * _buffer ;
        }
        CosNaming_BindingList ;

typedef struct
        {
        CosNaming_NamingContext         cxt ;
        CosNaming_Name                  rest_of_name ;
        }
        CosNaming_CannotProceed ;
```

```
typedef struct
        {
        unsigned long                   why ;
        CosNaming_Name                  rest_of_name ;
        }
        CosNaming_NotFound ;

/* values for CosNaming_NotFound.why */
#define CosNaming_NotFoundReason_missing_node          1
#define CosNaming_NotFoundReason_not_context           2
#define CosNaming_NotFoundReason_no_object             3
```

Description

Data structures for the naming service.

The naming service associates names with objects. A name is bound to an object relative to a *naming context*, which is itself an object. Since a naming context is an object, it can also be named.

The structures ***CosNaming::NameComponent***, ***CosNaming::Name***, ***CosNaming::Binding***, and ***CosNaming::BindingList*** are used as operations parameters.

The structures ***CosNaming::NotFound*** and ***CosNaming::CannotProceed*** are returned with exceptions.

Notes

1. A ***CosNaming::Name*** consists of a sequence of components. The most common interpretation is to view a ***CosNaming::Name*** as a path name, althought there are other possible semantics. In this interpretation, a naming context is roughly equivalent to a directory. When ***CosNaming::Name***s correspond to path names, the separator characters of the path names are not included in the ***CosNaming::Name*** structure, and all of the components except the last must be directories.

2. The value of ***CosNaming::BindingType*** is either ***CosNaming::BindingType::nobject*** or ***CosNaming::BindingType::ncontext***. In the most common interpretation, each components of a path name is either an object or a directory.

3. The CosNaming module defines two classes of interface: ***CosNaming::NamingContext*** and ***CosNaming::BindingIterator***.

4. The ***CosNaming*** module defines four exceptions:

Exception	Description	Value
CosNaming::NotFound	Unable to resolve the given name. A reason (***why***) and the unresolved portion of the name (***rest_of_name***) are returned.	yes
CosNaming::CannotProceed	The resolving context did not have permission to do a resolution. The context lacking permission (***cxt***) and the unresolved portion of the name (***rest_of_name***) are returned.	yes
CosNaming::InvalidName	The given name was not properly formed.	no
CosNaming::AlreadyBound	The given name was already bound to a different object. Note that an object can be bound to several names, but a name can be bound only to one object.	no

5. The ***CosNaming::CannotProceed*** exception indicated that the returned context did not have permission to resolve the rest of the name. However, in some situations, the caller might have sufficient permission to resolve the rest of the name directly by using ***CosNaming::resolve***.

6. A zero-length name is invalid, and can cause a ***CosNaming::InvalidName*** exception. An implementation is free to impose other restrictions on the values of a name.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, Chapter 3: "Naming Service Specification"

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.1.

CosNaming::BindingIterator - return additional bindings in a naming context

Synopsis - C

```
#include        <dsmcc.h>

CORBA_boolean       CosNaming_BindingIterator_next_one
                                ( CosNaming_BindingIterator     iterator1,
                                CosNaming_Binding               ** binding1,
                                CORBA_Environment               * ev ) ;

CORBA_boolean       CosNaming_BindingInterator_next_n
                                ( CosNaming_BindingIterator     iterator1,
                                unsigned long                   how_many,
                                CosNaming_BindingList           ** list1,
                                CORBA_Environment               * ev ) ;

void                CosNaming_BindingIterator_destroy
                                ( CosNaming_BindingIterator     iterator1,
                                CORBA_Environment               * ev ) ;
```

Arguments

iterator1	*(in)* the iterator returned from a ***list*** operation
binding1	*(out)* the next binding from the naming context
how_many	*(in)* the number of bindings to return
list1	*(out)* the next ***how_many*** bindings from the naming context
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Returns the next binding(s) from the naming context of a ***list*** operation.

The ***destroy*** operation discards any server-side storage associated with the iterator and destroys the iterator object.

Notes

(none)

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.7.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Access::Hist - version and time of a persistent object

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_Access_Hist_T ;

DSM_Access_Hist_T **  DSM_Access__get_Hist
                                ( CORBA_Object                  obj1,
                                CORBA_Environment               * ev ) ;

void                  DSM_Access__set_Hist
                                ( CORBA_Object                  obj1,
                                DSM_Access_Hist_T               * hist1,
                                CORBA_Environment               * ev ) ;
```

Arguments

obj1	*(in)* the object whose history is to be returned or modified
hist1	*(in)* the new update time and version of the object
ev	*(in/out)* CORBA environment

Returns

DSM_Access__get_Hist returns the ***Hist*** attribute of an object.
DSM_Access__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of an object.

Notes

1. The time is when the object was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_Access__get_Hist*** requires ***READER*** privilege. ***DSM_Access__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Access::Lock - status of read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_Access_Lock_T ;

DSM_Access_Lock_T *   DSM_Access__get_Lock ( CORBA_Object          obj1,
                                             CORBA_Environment     * ev ) ;

void                  DSM_Access__set_Lock ( CORBA_Object          obj1,
                                             DSM_Access_Lock_T     * lock1,
                                             CORBA_Environment     * ev ) ;
```

Arguments

obj1	*(in)* the object whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_Access__get_Lock returns ***DSM_Access_Lock_T***
DSM_Access__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of an object.

Notes

1. ***DSM_Access__get_Lock*** requires ***READER*** privilege. ***DSM_Access__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Access::Perms - access permission information for an object

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_Access_Perms_T ;

DSM_Access_Perms_T *    DSM_Access__get_Perms
                                ( CORBA_Object                  obj1,
                                CORBA_Environment               * ev ) ;

void                    DSM_Access__set_Perms
                                ( CORBA_Object                  obj1,
                                DSM_Access_Perms_T              * perms1,
                                CORBA_Environment               * ev1 ) ;
```

Arguments

obj1	*(in)* the object whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_Access__get_Perms returns ***DSM_Access_Perms_T***
DSM_Access__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of an object.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this object.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Access::Size - size of a persistent object

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong        DSM_Access__get_Size ( CORBA_Object          obj1,
                                             CORBA_Environment       * ev ) ;
```

Arguments

obj1	*(in)* the object whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of an object

Exceptions

(standard)

Description

Returns the size of an object's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Base::close - close a reference to an object

Synopsis - C

```
#include        <dsmcc.h>

void    DSM_Base_close ( CORBA_Object        obj1,
                         CORBA_Environment   * ev ) ;
```

Arguments

obj1	*(in)* the object whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to an object is no longer required. The object reference ***obj1*** is deleted, and it is no longer possible to communicate with the object.

Notes

1. If a parent ***DSM::Composite*** object is closed, its child objects are also closed as a result of this operation.

2. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

3. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::Base::destroy - destroy an object instance

Synopsis - C

```
#include        <dsmcc.h>

void    DSM_Base_destroy ( CORBA_Object       obj1,
                           CORBA_Environment  * ev ) ;
```

Arguments

obj1	*(in)* the object to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys an object, and releases the associated resources.

Notes

1. After this operation, the object no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::Composite::bind_subs - bind sub-objects to a composite object

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char aMajor ;  char aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        CosNaming_NameComponent             n ;
        DSM_Version                         rVersion ;
        CORBA_boolean                       required ;
        CORBA_Object                        obj ;
        }
        DSM_Composite_ChildBinding ;

typedef struct
        {
        unsigned long                   _maximum, _length ;
        DSM_Composite_ChildBinding      *_buffer ;
        }
        DSM_Composite_ChildBindings ;

typedef struct
        {
        unsigned long                   why ;
        CosNaming_Name                  rest_of_name ;
        }
        DSM_Composite_NotFound ;

void    DSM_Composite_bind_subs
                ( DSM_Composite                 composite1 ,
                CosNaming_Name                  * name1,
                DSM_Composite_ChildBindings     * bindings1 ;
                CORBA_Environment               * ev ) ;
```

Arguments

composite1	*(in)* the composite object to which the child objects are to be bound
name1	*(in)* path name to composite object
bindings1	*(in)* list of child object information
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::INV_NAME - path name is incorrectly formatted
DSM::Composite::NotFound - cannot resolve ***name1*** (returns ***DSM_Composite_NotFound***, the unresolved portion of ***name1*** with an explanation of the reason)
DSM::Composite::AlreadyBound - the name is already bound to another object

(also standard exceptions)

Description

Bind a set of required and optional objects to a composite binder.

Notes

1. This operation requires ***WRITER*** privilege.

2. The composite set consists of compatible versions of objects.

3. In the exception structure ***DSM_Composite_NotFound***, the member ***why*** is one of the following:
   DSM_Composite_NotFoundReason_missing_node
   DSM_Composite_NotFoundReason_not_context
   DSM_Composite_NotFound_Reason_not_object

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.3.3.

DSM::Composite::list_subs - list information about child object references

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char aMajor ;  char aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        CosNaming_NameComponent             n ;
        DSM_Version                         rVersion ;
        CORBA_boolean                       required ;
        }
        DSM_Composite_ChildInfo ;

typedef struct
        {
        unsigned long                   _maximum, _length ;
        DSM_Composite_ChildInfo         * _buffer ;
        }
        DSM_Composite_ChildInfos ;

typedef struct
        {
        unsigned long                   why ;
        CosNaming_Name                  rest_of_name ;
        }
        DSM_Composite_NotFound ;

void    DSM_Composite_list_subs
                ( DSM_Composite                     composite1 ,
                CosNaming_Name                      * name1,
                DSM_Composite_ChildInfos            ** info1 ;
                CORBA_Environment                   * ev ) ;
```

Arguments

composite1	*(in)* the composite object for which child information is sought
name1	*(in)* path name to composite object
info1	*(out)* child object information
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::INV_NAME - path name is incorrectly formatted
DSM::Composite::NoteFound - cannot resolve ***name1*** (returns ***DSM_Composite_NotFound***, the unresolved portion of ***name1*** with an explanation of the reason)

(also standard exceptions)

Description

Returns information about child objects.

Notes

1. This operation requires ***READER*** privilege.

2. The composite set consists of compatible versions of objects.

3. In the exception structure ***DSM_Composite_NotFound***, the member ***why*** is one of the following:
   DSM_Composite_NotFoundReason_missing_node
   DSM_Composite_NotFoundReason_not_context
   DSM_Composite_NotFound_Reason_not_object

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.3.2.

DSM::Composite::unbind_subs - remove sub-objects from composite object

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   why ;
        CosNaming_Name                  rest_of_name ;
        }
        DSM_Composite_NotFound ;

void    DSM_Composite_unbind_subs
                ( DSM_Composite                 composite1 ,
                CosNaming_Name                  * name1,
                CORBA_Environment               * ev ) ;
```

Arguments

composite1	*(in)* the composite object to which the child objects are to be bound
name1	*(in)* path name to composite object
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::INV_NAME - path name is incorrectly formatted
DSM::Composite::NoteFound - cannot resolve ***name1*** (returns ***DSM_Composite_NotFound***, the unresolved portion of ***name1*** with an explanation of the reason)

(also standard exceptions)

Description

Unbind all child objects from a composite object.

Notes

1. This operation requires ***WRITER*** privilege.

2. In the exception structure ***DSM_Composite_NotFound***, the member ***why*** is one of the following:
   DSM_Composite_NotFoundReason_missing_node
   DSM_Composite_NotFoundReason_not_context
   DSM_Composite_NotFound_Reason_not_object

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.3.4.

DSM::Config::inquire - check status of transaction

(to be supplied)

DSM::Config::wait - wait for transaction to complete

(to be supplied)

DSM::Directory - path traversal

Synopsis - C

```
#include        <dsmcc.h>

typedef char                                DSM_PathType ;
#define DSM_PathType_DEPTH                  'D'
#define DSM_PathType_BREADTH                'B'

typedef struct
        {
        CosNaming_NameComponent             name ;
        CORBA_boolean                       process ;
        }
        DSM_Step ;

typedef struct
        {
        unsigned long                       _maximum ;
        unsigned long                       _length ;
        DSM_Step                            *_buffer ;
        }
        DSM_PathSpec ;

typedef struct
        {
        unsigned long                       _maximum ;
        unsigned long                       _length ;
        CORBA_any                           *_buffer ;
        }
        DSM_PathValues ;
```

Description

Definitions of path specifications. These extend the concepts of ***CosNaming::Name***.

Notes

1. The ***DSM::Directory*** extensions to ***CosNaming*** include four operations: ***open***, ***close***, ***get***, and ***put***. The path traversal extensions relate to the ability of ***open***, ***get***, and ***put*** to resolve or otherwise to process multiple components of a path name in a single operation.

2. A ***CosNaming::Name*** consists of a sequence of name components, each of which is a <*name, binding type*> pair. ***A DSM::PathSpec*** expands the pair to a triple: <*name, binding type, process*>. The ***process*** is a boolean (*true* or *false*) value used in path traversal at each step (or component) to match input or output lists to path steps (components). If ***process = 1***, then the current step corresponds to an input or to an output value, and if ***process = 0***, then the current step does not correspond to an input or output value. The number of data values (objects, attributes, etc) in the list argument to an ***open***, ***get***, or ***put*** operation is equal to the number of steps for which ***process = 1*** in the ***DSM::PathSpec*** argument.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.1.

DSM::Directory::bind - bind object reference to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_bind
              ( DSM_Directory          dir1,
              CosNaming_Name           * name1,
              CORBA_Object             obj1,
              CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the binding is to occur
name1	*(in)* the name to which the object is to be bound
obj1	*(in)* the object to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a naming context.

Notes

1. Requires ***WRITER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.5.

DSM::Directory::bind_context - bind directory to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_bind_context
              ( DSM_Directory              dir1,
              CosNaming_Name               * name1,
              CosNaming_NamingContext      dir2,
              CORBA_Environment            * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the binding is to occur
name1	*(in)* the name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a naming context. (Binds ***dir2*** to ***name1*** within ***dir1***).

Notes

1. Requires ***WRITER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.6.

DSM::Directory::close - close a reference to a directory

Synopsis - C

```
#include        <dsmcc.h>

void    DSM_Directory_close ( DSM_Directory    dir1,
                              CORBA_Environment * ev ) ;
```

Arguments

dir1	*(in)* the directory whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a directory is no longer required. The object reference ***dir1*** is deleted, and it is no longer possible to communicate with the directory.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::Directory::destroy - destroy a directory

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_Directory_destroy ( DSM_Directory          dir1,
                                 CORBA_Environment      * ev ) ;
```

Arguments

dir1	*(in)* the directory to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a directory, and releases the associated resources.

Notes

1. After this operation, the directory no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::Directory::get - return attribute values bound to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_get
              ( DSM_Directory          dir1,
              DSM_PathType             path_type,
              DSM_PathSpec             *path_spec,
              DSM_PathValues           ** path_values,
              CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory from which attribute values are sought
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(out)* attribute values returned
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Returns attribute values corresponding to the given path specification.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is returned.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be returned, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::Directory::Hist - version and time of a directory

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_Directory_Hist_T ;

DSM_Directory_Hist_T *      DSM_Directory__get_Hist
                                ( DSM_Directory             dir1,
                                CORBA_Environment           * ev ) ;

void                        DSM_Directory__set_Hist
                                ( DSM_Directory             dir1,
                                DSM_Directory_Hist_T        * hist1,
                                CORBA_Environment           * ev ) ;
```

Arguments

dir1	*(in)* the directory whose history is to be returned or modified
hist1	*(in)* the new update time and version of the directory
ev	*(in/out)* CORBA environment

Returns

DSM_Directory__get_Hist returns the ***Hist*** attribute of a directory.
DSM_Directory__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a directory.

Notes

1. The time is when the directory was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_Directory__get_Hist*** requires ***READER*** privilege. ***DSM_Directory__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Directory::list - return list of bindings

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_list
                ( DSM_Directory              dir1,
                unsigned long                how_many,
                CosNaming_BindingList        ** binding_list,
                CosNaming_BindingIterator    * binding_iterator,
                CORBA_Environment            * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) from which bindings are to be listed
how_many	*(in)* the number of bindings to return initially
binding_list	*(out)* the initial list of bindings
binding_iterator	*(out)* the iterator object used to obtain additional bindings
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Returns a list of bindings from a naming context.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.6.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.3.

DSM::Directory::Lock - status of directory read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_Directory_Lock_T ;

DSM_Directory_Lock_T *      DSM_Directory__get_Lock
                                ( DSM_Directory             dir1,
                                CORBA_Environment           * ev ) ;

void                        DSM_Directory__set_Lock
                                ( DSM_Directory             dir1,
                                DSM_Directory_Lock_T        * lock1,
                                CORBA_Environment           * ev ) ;
```

Arguments

dir1	*(in)* the directory whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_Directory__get_Lock returns ***DSM_Directory_Lock_T***
DSM_Directory__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a directory.

Notes

1. ***DSM_Directory__get_Lock*** requires ***READER*** privilege. ***DSM_Directory__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Directory::new_context - create a new directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_new_context
              ( DSM_Directory              dir1,
              DSM_Directory                * new_directory,
              CORBA_Environment            * ev1 ) ;
```

Arguments

dir1	*(in)* any directory (context) on the server where the new context is to be created
new_directory	*(in)* the new directory
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Creates a new directory on the same server as an existing directory.

Notes

1. Requires ***OWNER*** privilege.
2. The new directory is not bound to any context.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.4.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.10.

DSM::Directory::open - resolve the objects of a path

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_Directory_open
                ( DSM_Directory         dir1,
                DSM_PathType            path_type,
                DSM_PathSpec            * path_spec,
                DSM_ObjRefs             ** resolved_references,
                CORBA_Environment       * ev1 ) ;
```

Arguments

dir1	*(in)* the directory to be used as context for the path specification
path_type	*(in)* the method of path traversal
path_spec	*(in)* the path to be resolved
resolved_references	*(out)* the resolved objects from the path
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

DSM::OPEN_LIMIT - attempt to surpass the maximum allowed number of object references

(also standard exceptions)

Description

Finds the objects associated with the names in a given path.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*directory*>, <*object or directory*>. Each node for which ***process = 1*** is resolved to an object reference in the output list.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to objects within the naming context. The objects (which may be directory objects) are all within the given directory, and each (for which ***process = 1***) is resolved to an object reference for the output list.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The function of this operation is similar to that of ***resolve***. However, ***list*** allows multiple simultaneous resolutions of names.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.13.

DSM::Directory::Perms - access permission information for a directory

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_Directory_Perms_T ;

DSM_Directory_Perms_T *     DSM_Directory__get_Perms
                                ( DSM_Directory             dir1,
                                CORBA_Environment           * ev ) ;

void                        DSM_Directory__set_Perms
                                ( DSM_Directory             dir1,
                                DSM_Directory_Perms_T       * perms1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

dir1	*(in)* the directory whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_Directory__get_Perms returns ***DSM_Directory_Perms_T***
DSM_Directory__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a directory.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this directory.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Directory::put - bind attribute values to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_put
              ( DSM_Directory          dir1,
              DSM_PathType             path_type,
              DSM_PathSpec             *path_spec,
              DSM_PathValues           * path_values,
              CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory to which attribute values are to be bound
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(in)* attribute values
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Sets attribute values corresponding to the given path specification.

Notes

1. Requires ***WRITER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is bound.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be bound, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::Directory::rebind - rename an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_rebind
              ( DSM_Directory          dir1,
              CosNaming_Name           * new_name,
              CORBA_Object             obj1,
              CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the name is to be changed
new_name	*(in)* the new name for the object
obj1	*(in)* the object to be renamed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Renames an object within a given context. This operation replaces an existing name binding in the context.

Notes

1. Requires ***WRITER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.7.

DSM::Directory::rebind_context - rename a directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_rebind_context
              ( DSM_Directory              dir1,
              CosNaming_Name               * new_name,
              CosNaming_NamingContext      dir2,
              CORBA_Environment            * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the binding is to occur
name1	*(in)* the new name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Binds a new name to an object within a naming context. (Binds ***dir2*** to ***name1*** within ***dir1***). An existing binding for the same name in the context is destroyed.

Notes

1. Requires ***WRITER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.8.

DSM::Directory::resolve - return object reference for given name

Synopsis - C

```
#include      <dsmcc.h>

CORBA_Object  DSM_Directory_resolve
                    ( DSM_Directory          dir1,
                    CosNaming_Name           * name1,
                    CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the binding is to sought
name1	*(in)* the name for which the binding is to be located
ev	*(in/out)* CORBA environment

Returns

an object reference for the binding in the given context

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Given a name, returns an object reference for an object.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.2.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.4.

DSM::Directory::Size - size of a directory

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong        DSM_Directory__get_Size ( DSM_Directory        dir1,
                                CORBA_Environment               * ev ) ;
```

Arguments

dir1	*(in)* the directory whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a directory

Exceptions

(standard)

Description

Returns the size of an directory's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Directory::unbind - remove name from an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Directory_unbind
              ( DSM_Directory          dir1,
              CosNaming_Name           * name1,
              CORBA_Environment        * ev1 ) ;
```

Arguments

dir1	*(in)* the directory (context) in which the binding exists
name1	*(in)* the name for which the binding is to be removed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Remove the binding for a name within a context.

Notes

1. Requires ***WRITER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.3.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.9.

DSM::Download::alloc - allocate memory buffers for a download

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Download_alloc
              ( DSM_Download          download_obj,
              unsigned short          module_id,
              CORBA_Object            write_buffer,
              CORBA_Object            read_buffer,
              CORBA_Environment       * ev1 ) ;
```

Arguments

download_obj	*(in)* the object to be downloaded
module_id	*(in)* the module of the object to be downloaded
write_buffer	*(in)* an application-provided buffer object
read_buffer	*(out)* a system-provided buffer object
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_MODULE_ID - invalid module identifier
(also standard exceptions)

Description

Specifies or allocates a buffer for a module to be downloaded.

If ***write_buffer*** is not NIL, then the application assigns the buffer and ***read_buffer*** is returned NIL. If ***write_buffer*** is NIL, then a buffer object is returned in ***read_buffer***.

Notes

1. Requires ***READER*** privilege.

2. The ***module_id*** is taken from a previous ***DSM::Download::info*** operation.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.1.4.

DSM::Download::cancel - cancel a download in progress

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        unsigned short                  * _buffer ;
        }
        DSM_ModuleList ;

void    DSM_Download_cancel
                ( DSM_Download                  download_obj,
                DSM_ModuleList                  * module_list,
                CORBA_Environment               * ev1 ) ;
```

Arguments

download_obj	*(in)* the object being downloaded
module_list	*(in)* the list of modules for which download is to be cancelled
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_MODULE_ID - invalid module identifier
DSM::TIMEOUT - the maximum time for a transaction has been exceeded
(also standard exceptions)

Description

Cancels a download in progress for specific modules.

Notes

1. Requires ***READER*** privilege.

2. The ***module_id***s within ***module_list*** are taken from previous ***DSM::Download::info*** and ***DSM::Download::start*** operations.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.1.6.

DSM::Download::info - obtain information about prerequisite download modules

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        unsigned char                   * _buffer ;
        }
        module_info_t ;

typedef struct
        {
        unsigned short                  moduleId ;
        unsigned char                   moduleVersion ;
        unsigned long                   moduleSize ;
        module_info_t                   moduleInfoBytes ;
        }
        DSM_ModuleInfo ;

typedef struct
        {
        unsigned long                           _maximum ;
        unsigned long                           _length ;
        DSM_ModuleInfo                          * _buffer ;
        }
        DSM_ModuleInfoList ;

void    DSM_Download_info
                ( DSM_Download                  download_obj,
                DSM_ModuleInfoList              ** info_list,
                CORBA_Environment               * ev1 ) ;
```

Arguments

download_obj	*(in)* the object to be downloaded
info_list	*(in)* information about the modules in ***download_obj***
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard exceptions)

Description

Provides information about the modules of a download object.

Notes

1. The data of a module are organized into modules, which in turn are divided into blocks. An application may require some or all of the modules of a download object: this is an implementation-dependent decision.

2. The ***moduleInfoBytes*** may be used to select among the available modules for downloading, based on implementation-dependent criteria. Once the modules are selected, ***DSM::Download::start*** may be used to initiate the download operation.

3. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.1.3.

DSM::Download::start - start download and transfer modules to client

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        unsigned short                  * _buffer ;
        }
        DSM_ModuleList ;

void    DSM_Download_start
                ( DSM_Download                  download_obj,
                DSM_ModuleList                  * module_list,
                CORBA_Environment               * ev1 ) ;
```

Arguments

download_obj	*(in)* the object being downloaded
module_list	*(in)* the list of modules for which download is to be started
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_MODULE_ID - invalid module identifier
DSM::TIMEOUT - the maximum time for a transaction has been exceeded
DSM::MPEG_DELIVERY - the server was unable to deliver an object over the MPEG stream
(also standard exceptions)

Description

Begins a download operation for the indicated modules.

Notes

1. Requires ***READER*** privilege.

2. The ***module_id***s within ***module_list*** are taken from a previous ***DSM::Download::info*** operation.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.1.5.

DSM::DownloadSI::cancel - cancel download in progress

Synopsis - C

```
#include      <dsmcc.h>

typedef struct
        {
        unsigned short                  moduleId ;
        unsigned short                  blockNumber ;
        unsigned char                   downloadCancelReason ;
        struct
                {
                unsigned long           _maximum ;
                unsigned long           _length ;
                unsigned char           * _buffer ;
                }
                privateDataBytes ;
        }
        DSM_DownloadSI_CancelRequest ;

void    DSM_DownloadSI_cancel
                ( DSM_DownloadSI                download_obj,
                DSM_DownloadSI_CancelRequest    * cancel_request,
                CORBA_Environment               * ev1 ) ;
```

Arguments

download_obj	*(in)* the object being downloaded
cancel_request	*(in)* information about the cancellation
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_MODULE_ID - invalid module identifier
DSM::TIMEOUT - the maximum time for a transaction has been exceeded
(also standard exceptions)

Description

Cancels a download operation.

Notes

1. Requires ***READER*** privilege.

2. The ***privateDataBytes*** allow an implementation-dependent mechanism for providing more information about the cancellation.

3. The ***downloadCancelReason*** is taken from the following values:

Reason	Value	Sender	Description
	0x00		ISO/IEC 13818-6 reserved
rsnScenarioTimeout	0x01	Server, Client	Timer ***tCDownloadScenario*** expired
rsnInsufMem	0x02	Server, Client	Insufficient memory
rsnAuthDenied	0x03	Server	Download authorization denied
rsnFatal	0x04	Server, Client	Fatal error
rsnInfoRequestError	0x05	Server	The Download Server cannot accommodate the requested maximum ***blockSize*** or ***bufferSize***
rsnCompatError	0x06	Server	The Download Server cannot determine an appropriate image from the ***compatibilityDescriptor*** provided by the client
rsnUnreliableNetwork	0x07	Server	The Client indicated protocol settings for a reliable download but Download Server has determined that the level of service provided by the network layer is unreliable
rsnInvalidData	0x08	Client	The Client received data which did not match the description in the ***moduleInfoByte*** fields in the ***DownloadInfoResponse*** or ***DownloadInfoIndication*** message
rsnInvalidBlock	0x09	Client	The Client received a block with an invalid ***moduleId*** or ***blockNumber***
rsnInvalidVersion	0x0a	Client	The Client received a block with an unexpected value of the ***moduleVersion*** field
rsnAbort	0x0b	Client	The Client aborts the download scenario in progress
rsnRetrans	0x0c	Server, Client	The sender has reached the maximum allowed number of retransmissions

Reason	Value	Sender	Description
rsnBadBlockSize	0x0d	Client	The Client cannot support the selected value for ***blockSize***
rsnBadWindow	0x0e	Client	The Client cannot support the selected value of ***windowSize***
rsnBadAckPeriod	0x0f	Client	The Client cannot support the selected value for ***ackPeriod***
rsnBadWindowTimer	0x10	Client	The Client cannot support the selected value for ***tCDownloadWindow***
rsnBadScenarioTimer	0x11	Client	The Client cannot support the selected value for ***tCDownloadScenario***
rsnBadCapabilities	0x12	Client	The Client cannot parse the ***compatibilityDescriptor***
rsnBadModuleTable	0x13	Client	The Client cannot parse the module table
	0x14-0xef		ISO/IEC 13818-6 reserved
	0xf0-0xff		Private use

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.5 and 7.3.5.

DSM::DownloadSI::deinstall - unbind download configuration from server

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_DownloadSI_deinstall
              ( DSM_DownloadSI                  download_obj,
              DSM_CompatibilityDescriptor       * compatibility_descriptor,
              CORBA_Environment                 * ev1 ) ;
```

Arguments

download_obj	*(in)* the download object bound to the configuration
compatibility_descriptor	*(in)* the download configuration to be unbound
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_COMPAT_INFO - unrecognized compatibility descriptor
(also standard exceptions)

Description

Unbinds a download configuration from the server. The compatibility descriptor specifies which configuration is to be unbound.

Notes

1. Requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.5.

DSM::DownloadSI::info - obtain download information and negotiate parameters

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_DownloadSI_info
              ( DSM_DownloadSI          download_obj,
              DSM_InfoRequest           * request,
              DSM_InfoResponse          ** response,
              CORBA_Environment         * ev1 ) ;
```

Arguments

download_obj	*(in)* the download object
request	*(in)* the requested parameters
response	*(out)* the reply parameters
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_COMPAT_INFO - unrecognized compatibility descriptor
DSM::BUF_SIZE - the requested buffer size is unacceptable to the server
DSM::BLOCK_SIZE - the requested block size is unacceptable to the server
(also standard exceptions)

Description

Obtain module information and negotiate flow-control parameters for a download operation.

Notes

1. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.5.

DSM::DownloadSI::install - bind download configuration to server

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short                  moduleId ;
        unsigned char                   moduleVersion ;
        unsigned long                   moduleSize ;
        struct
                {
                unsigned long           _maximum ;
                unsigned long           _length ;
                unsigned char           *_buffer ;
                }
                moduleInfoBytes ;
        }
        DSM_moduleInfo ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        DSM_moduleInfo                  *_buffer ;
        }
        DSM_ModuleInfoList ;

typedef struct
        {
        unsigned short                  aModuleId ;
        DSM_CosNaming_Name              n ;
        }
        DSM_DownloadSI_ModuleInstallInfo ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        DSM_DownloadSI_ModuleInstallInfo *_buffer ;
        }
        DSM_DownloadSI_ModuleInstallList ;
```

```
void    DSM_DownloadSI_install
            ( DSM_DownloadSI                download_obj,
            DSM_CompatibilityDescriptor     * compatibility_descriptor,
            DSM_ModuleInfoList              * module_information,
            DSM_ModuleInstallList           * path_information
            CORBA_Environment               * ev1 ) ;
```

Arguments

download_obj	*(in)* the download object to bind to the configuration
compatibility_descriptor	*(in)* the download configuration to be bound
module_information	*(in)* information about the modules
path_information	*(in)* paths to objects containing download data
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_COMPAT_INFO - unrecognized compatibility descriptor
DSM::BAD_MODULE_INFO - incorrect download install configuration
DSM::INV_NAME - path name is incorrectly formatted
DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object
(also standard exceptions)

Description

Binds a download configuration to a server.

Notes

1. Requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.5.

DSM::DownloadSI::proceed - transfer download data blocks

Synopsis - C

```
#include      <dsmcc.h>

typedef struct
        {
        unsigned short              moduleId ;
        unsigned short              blockNumber ;
        unsigned char               downloadReason ;
        }
        DSM_DownloadSI_DataRequest ;

void    DSM_DownloadSI_proceed
              ( DSM_DownloadSI                  download_obj,
              DSM_DownloadSI_DataRequest        * request,
              CORBA_Environment                 * ev1 ) ;
```

Arguments

download_obj	*(in)* the download object
request	*(in)* the module and block to be downloaded
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::BAD_MODULE_ID - unrecognized compatibility descriptor
DSM::TIMEOUT - the maximum allowed time for a transaction has been exceeded
DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream
(also standard exceptions)

Description

Transfer a series of ***DownloadDataBlock***s to the Client.

Notes

1. Requires ***READER*** privilege.

2. In ***DSM::DownloadSI::DataRequest***, the ***moduleId*** and ***blockNumber*** identify the next block to be transferred. The total number of blocks to transfer is determined by the negotiated window size. The value of ***downloadReason*** is taken from the following:

downloadReason	Value	Description

downloadReason	Value	Description
	0x00	ISO/IEC 13818-6 reserved
rsnStart	0x01	start request for download
rsnAckCont	0x02	acknowledge reception of previous blocks, and request continuation of transmission from indicated block
rsnNakRetransBlock	0x03	request retransmission of blocks starting from indicated block because blocks are missing
rsnNakRetransWindow	0x04	request retransmission of blocks starting from indicated block because timer *tCDownloadWindow* expired
rsnEnd	0x05	end download because all blocks are successfully received
	0x06-0xef	ISO/IEC 13818-6 reserved
	0xf0-0xff	private use

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.5.

DSM::Event::notify - obtain event data from stream event descriptor

(to be supplied)

DSM::Event::subscribe - subscribe to receive an MPEG stream event

(to be supplied)

DSM::Event::unsubscribe - end subscription to an MPEG stream event

(to be supplied)

DSM::File::close - close a reference to a file

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_File_close ( DSM_File        file1,
                          CORBA_Environment              * ev ) ;
```

Arguments

file11	*(in)* the file whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a file is no longer required. The object reference ***file1*** is deleted, and it is no longer possible to communicate with the file.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::File::destroy - destroy a file

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_File_destroy ( DSM_File              file1,
                            CORBA_Environment     * ev ) ;
```

Arguments

file1	*(in)* the file to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a file, and releases the associated resources.

Notes

1. After this operation, the file no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::File::Hist - version and time of a file

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_File_Hist_T ;

DSM_File_Hist_T *       DSM_File__get_Hist
                                ( DSM_File                      file1,
                                CORBA_Environment               * ev ) ;

void                    DSM_File__set_Hist
                                ( DSM_File                      file1,
                                DSM_File_Hist_T                 * hist1,
                                CORBA_Environment               * ev ) ;
```

Arguments

file1	*(in)* the file whose history is to be returned or modified
hist1	*(in)* the new update time and version of the file
ev	*(in/out)* CORBA environment

Returns

DSM_File__get_Hist returns the ***Hist*** attribute of a file
DSM_File__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a file.

Notes

1. The time is when the file was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_File__get_Hist*** requires ***READER*** privilege. ***DSM_File__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::File::Lock - status of file read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_File_Lock_T ;

DSM_File_Lock_T *       DSM_File__get_Lock ( DSM_File                   file1,
                                             CORBA_Environment          * ev ) ;

void                    DSM_File__set_Lock ( DSM_File                   file1,
                                             DSM_File_Lock_T            * lock1,
                                             CORBA_Environment          * ev ) ;
```

Arguments

file1	*(in)* the file whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_File__get_Lock returns ***DSM_File_Lock_T***
DSM_File__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a file.

Notes

1. ***DSM_File__get_Lock*** requires ***READER*** privilege. ***DSM_File__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::File::Perms - access permission information for a file

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_File_Perms_T ;

DSM_File_Perms_T *      DSM_File__get_Perms
                                ( DSM_File                  file1,
                                CORBA_Environment           * ev ) ;

void                    DSM_Access__set_Perms
                                ( DSM_File                  file1,
                                DSM_File_Perms_T            * perms1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

file1	*(in)* the file whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_File__get_Perms returns ***DSM_File_Perms_T***
DSM_File__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a file.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this file.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::File::read - random access read from a file

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_File_read  ( DSM_File                file1,
                          DSM_u_longlong          offset1,
                          unsigned long           size1,
                          CORBA_boolean           reliable1,
                          DSM_opaque              ** data1,
                          CORBA_Environment       * ev ) ;
```

Arguments

file1	*(in)* the file to be read
offset1	*(in)* the offset in the file from which the read operation is to begin
size1	*(in)* the number of octets to read from the file
reliable1	*(in)* indicates whether operation is to be re-tried in case of timeout or error
data1	*(out)* the data returned from the file
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::INV_OFFSET - specified offset is greater than file size
DSM::INV_SIZE - size to be read exceeds network, server, or client limitations
DSM::READ_LOCKED - read operations are temporarily prevented
(also standard exceptions)

Description

Reads information from a file.

Notes

1. This operation requires ***READER*** privilege.

2. If the offset is within the boundaries of the file, then a length which would cause a read beyond the end of the file does not generate and exception.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.4.3.

DSM::File::Size - size of a file

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong      DSM_File__get_Size ( DSM_File                   file1,
                                         CORBA_Environment          * ev ) ;
```

Arguments

file1	*(in)* the file whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a file

Exceptions

(standard)

Description

Returns the size of an file's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::File::write - random access write to a file

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_File_write ( DSM_File                file1,
                          DSM_u_longlong          offset1,
                          unsigned long           size1,
                          DSM_opaque              * data1,
                          CORBA_Environment       * ev ) ;
```

Arguments

file1	*(in)* the file to be written
offset1	*(in)* the offset in the file at which the write operation is to begin
size1	*(in)* the number of octets to write to the file
data1	*(out)* the data to be written to the file
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::INV_OFFSET - specified offset is greater than file size
DSM::INV_SIZE - size to be written exceeds network, server, or client limitations
DSM::WRITE_LOCKED - write operations are temporarily prevented
(also standard exceptions)

Description

Writes information to a file.

Notes

1. This operation requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.4.4.

DSM::First::root - obtain service gateway object

Synopsis - C

```
#include      <dsmcc.h>

CORBA_Object  DSM_First_root
                    ( DSM_First                    first1,
                    CORBA_Environment              * ev ) ;
```

Arguments

first1	*(in)* the first DSM-CC object
ev	*(in/out)* CORBA environment

Returns

the object reference to the ***ServiceGateway*** for the current session

Exceptions

(standard exceptions)

Description

Returns the object reference of the ***ServiceGateway*** for the current session.

Notes

1. This operation requires no privilege.
2. The ***DSM::First*** object is returned by ***DSM::Session::attach*** as a member of the output sequence of resolved references.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.8.2.

DSM::First::service - obtain primary service object

Synopsis - C

```
#include        <dsmcc.h>

CORBA_Object    DSM_First_service
                        ( DSM_First                     first1,
                        CORBA_Environment               * ev ) ;
```

Arguments

first1	*(in)* the first DSM-CC object
ev	*(in/out)* CORBA environment

Returns

the object reference to the current session's primary service object

Exceptions

(standard exceptions)

Description

Returns the object reference of the current session's primary service object

Notes

1. This operation requires no privilege.
2. The ***DSM::First*** object is returned by ***DSM::Session::attach*** as a member of the output sequence of resolved references.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.8.3.

DSM::Interfaces::check - verify coherence of interface with repository

Synopsis - C

```
#include        <dsmcc.h>

typedef unsigned long           DSM_IFKind ;

void    DSM_Interfaces_check
                        ( DSM_Interfaces                intf1,
                        DSM_IFKind                      ifkind1,
                        CORBA_Environment               * ev ) ;
```

Arguments

first1	*(in)* the interfaces service for a service domain
ifkind1	*(in)* identification of the interface to be checked
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_REF_TYPE - the referenced object or type does not exist in this context *(standard exceptions)*

Description

Verifies the coherency of interface ***ifkind1*** against the service domain's repository. The repository must assure that all interfaces defined with it represent a valid collection of IDL definitions.

Notes

1. This operation requires ***MANAGER*** privilege.

2. Among other things, this operation checks that all interfaces defined with ***ifkind1*** exist, that there are no duplicate operation names or other name collisions, that all parameters have known types, etc.

3. The interfaces service ***DSM::Interfaces*** may be found by searching the directory of the service domain. It has no standard location. (The service may not exist in all service domains.)

4. The ***DSM::IFKind*** is taken from the ***DSM::IntfCode*** structure, which is returned from ***DSM::Interfaces::define*** when the interface is defined, or from ***DSM::Interfaces::show*** when the repository is interrogated.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.6.7.

DSM::Interfaces::define - define an object interface to the system

Synopsis - C

```
#include        <dsmcc.h>

typedef unsigned long                   DSM_IFKind ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        DSM_IFKind                      * _buffer ;
        }
        DSM_IFKindList ;

typedef struct
        {
        unsigned long                   anIFKind ;
        char                            * kind ;
        char                            * repositoryId ;
        DSM_IFKindList                  includes ;
        }
        DSM_IntfCode ;

typedef DSM_opaque                      DSM_Interfaces_ReferenceData ;
typedef char                            * DSM_Interfaces_InterfaceDef ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        DSM_opaque                      * _buffer ;
        }
        DSM_TypeCode_List ;

void    DSM_Interfaces_define
                        ( DSM_Interfaces                intf1,
                        DSM_Interfaces_ReferenceData    * reference_data,
                        DSM_Interfaces_InterfaceDef     idl_description ;
                        DSM_IntfCode                    ** interface_code ;
                        DSM_TypeCodeList                ** type_codes ;
                        CORBA_Environment               * ev ) ;
```

Arguments

first1	*(in)* the interfaces service for a service domain
ifkind1	*(in)* user-defined reference data for this interface definition

idl_description	*(in)* the CORBA interface definition language (IDL) which describes the new interface class
interface_code	*(out)* the returned interface code assigned by the repository
type_codes	*(out)* type codes defined in this interface
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_REF_TYPE - the referenced object or type does not exist in this context
DSM::PREV_DEFINED - a data type or interface is already defined in this context
DSM::ILLEGAL_SYNTAX - the description which contained illegal IDL syntax
(also standard exceptions)

Description

Defines a new interface class.

Notes

1. This operation requires ***MANAGER*** privilege.

2. The interfaces service ***DSM::Interfaces*** may be found by searching the directory of the service domain. It has no standard location. (The service may not exist in all service domains.)

3. The ***DSM::IntfKind*** is taken from the ***DSM::IntfCode*** structure, which is returned from ***DSM::Interfaces::define*** when the interface is defined, or from ***DSM::Interfaces::show*** when the repository is interrogated.

4. If ***DSM::AccessRole*** is not explicitly defined for the new interface, then ***READER*** is assumed.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.6.6.

Object Management Group (OMG), *Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1996.

DSM::Interfaces::show - show interface definition information

Synopsis - C

```
#include        <dsmcc.h>

typedef unsigned long                   DSM_IFKind ;

typedef struct
        {
        unsigned long                   anIFKind ;
        char                            * kind ;
        char                            * repositoryId ;
        DSM_IFKindList                  includes ;
        }
        DSM_IntfCode ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        DSM_opaque                      * _buffer ;
        }
        DSM_TypeCode_List ;

void    DSM_Interfaces_show
                        ( DSM_Interfaces                intf1,
                        char                            * kind_string,
                        DSM_IFKind                      kind_value,
                        char                            ** idl_description,
                        DSM_IntfCode                    ** interface_code,
                        DSM_TypeCodeList                ** typecode_list,
                        CORBA_Environment               * ev ) ;
```

Arguments

first1	*(in)* the interfaces service for a service domain
kind_string	*(in)* kind of interface to be shown, such as *"DSM::Stream"*
kind_value	*(in)* kind of interface to be shown, such as *ik_Stream*
idl_description	*(out)* the CORBA interface definition language (IDL) which describes the interface class
interface_code	*(out)* the returned interface code assigned by the repository
type_codes	*(out)* type codes defined in this interface
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Show an interface definition, with interface code and type codes.

Notes

1. This operation requires ***READER*** privilege.

2. The interfaces service ***DSM::Interfaces*** may be found by searching the directory of the service domain. It has no standard location. (The service may not exist in all service domains.)

3. Either ***kind_string*** must be specified (to look up by name), or ***kind_value*** (to look up by code).

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.6.5.

Object Management Group (OMG), *Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1996.

DSM::Interfaces::undefine - remove object interface definition from the system

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_Interfaces_undefine
                        ( DSM_Interfaces                intf1,
                        DSM_Interfaces_ReferenceData    * reference_data,
                        DSM_IFKindList                  ** used_by,
                        CORBA_Environment               * ev ) ;
```

Arguments

first1	*(in)* the interfaces service for a service domain
reference_data	*(in)* the reference data used to define the interface
used_by	*(out)* a list of interfaces which use this interface
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard exceptions)

Description

Remove an interface definition from the repository.

Notes

1. This operation requires ***MANAGER*** privilege.

2. The interfaces service ***DSM::Interfaces*** may be found by searching the directory of the service domain. It has no standard location. (The service may not exist in all service domains.)

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.6.6.

DSM::Kind::has_a - determine whether object supports an interface

Synopsis - C

```
#include        <dsmcc.h>

typedef unsigned long                   DSM_IFKind ;

void    DSM_Kind_has_a
                        ( CORBA_Object                  obj1,
                        DSM_IFKind                      kind1,
                        CORBA_boolean                   yes_or_no,
                        CORBA_Environment               * ev ) ;
```

Arguments

obj1	*(in)* any object
kind1	*(in)* an interface kind
yes_or_no	*(out)* whether object supports interface
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard exceptions)

Description

Determines whether an object includes (inherits) a given interface.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.10.2.

DSM::Kind::is_a - show interfaces supported by an object

Synopsis - C

```
#include        <dsmcc.h>

typedef unsigned long                   DSM_IFKind ;

typedef struct
        {
        unsigned long                   anIFKind ;
        char                            * kind ;
        char                            * repositoryId ;
        DSM_IFKindList                  includes ;
        }
        DSM_IntfCode ;

void    DSM_Kind_Is_a
                        ( CORBA_Object                  obj1,
                        DSM_IntfCode                    ** interface_code,
                        CORBA_Environment               * ev ) ;
```

Arguments

obj1	*(in)* any object
interface_code	*(out)* returned interface code for the object
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard exceptions)

Description

Returns the interface code for an object.

Notes

1. This operation requires ***READER*** privilege.

2. Inherited interfaces are provided with the ***includes*** member of the interface code.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.10.3.

DSM::LifeCycle::create - create an object reference of specified kind

Synopsis - C

```
void    DSM_LifeCycle_create
                    ( DSM_LifeCycle          lifecycle1,
                    char                     * kind1,
                    DSM_Version              * version1,
                    CORBA_Object             * new_object1,
                    CORBA_Environment        * ev ) ;
```

Arguments

lifecycle1	*(in)* the lifecycle object to create the new object
kind1	*(in)* the kind of the new object
version1	*(in)* the version of the new object
new_object1	*(out)* the new object
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Creates a new object of the specified kind.

Notes

1. This operation requires ***OWNER*** privilege.

2. This definition differs from that of the standard. In the standard, the type of ***new_object1*** is ***CORBA::IOP::IOR***. Since this results in a number of incompatibilities and problems, this implementation outputs a CORBA ***Object*** type value. For further discussion, please refer to the chapter of this document, "Compliance with Applicable Standards," above.

3. The ***kind1*** string is of the form "<*module*>::<*interface*>". (For example, "*DSM::Stream*".)

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.9.1.

DSM::Security::authenticate - request authentication with password or key

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                                    * aPassword ;
        DSM_opaque                              authData ;
        }
        DSM_AuthRequest ;

typedef DSM_AuthRequest                         DSM_AuthRequest_T ;

void    DSM_Security_authenticate
                        ( CORBA_Object                  obj1,
                        DSM_AuthRequest_T               * authorization_info,
                        CORBA_Environment               * ev ) ;
```

Arguments

obj1	*(in)* the object requiring authentication
authorization_info	*(in)* password or other security information
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard exceptions)

Description

Requests authentication to an object.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.7.1.

DSM::ServiceGateway::attach - attach to a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        unsigned char                   * _buffer ;
        }
        DSM_Session_ServiceDomain ;

typedef DSM_opaque                      DSM_UserContext ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_ServiceGateway_attach
                ( DSM_ServiceGateway            gateway1,
                DSM_Session_ServiceDomain       * server_id,
                CosNaming_Name                  * path_name,
                DSM_UserContext                 * saved_context,
                DSM_ObjRefs                     ** resolved_refs,
                CORBA_Environment               * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway for the session
server_id	*(in)* server identifier (optional)
path_name	*(in)* path name to resolve (optional)
saved_context	*(in)* previous application context (optional)
resolved_refs	*(out)* resolved object references (see Notes, below)
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::OPEN_LIMIT - attempt to surpass the maximum allowable number of active object references

DSM::NO_AUTH - the end-user has not provided the correct authentication in the request

DSM::UNK_USER - the end user is unknown to the service domain
DSM::SERVICE_XFR -a resolve operation was unsuccesful (an alternate service domain location is provided)
DSM::BAD_COMPAT_INFO - an unrecognized compatibility descriptor was provided
DSM::NO_RESUME - the previous application saved state cannot be recovered
DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object
DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path
DSM::INV_NAME - a path name is incorrectly formatted
(also standard exceptions)

Description

Attach to a service gateway domain, establishing a session context.

Notes

1. Requires ***READER*** privilege.

2. Either ***server_id*** or ***path_name*** (or both) must be provided. If both are given, then ***server_id*** specifies the server, and ***path_name*** gives the path to a service gateway and (optionally) to a first service.

3. The ***server_id*** must be in NSAP address format. For an interactive session, it is the globally unique server network address. For a broadcast carousel session, it is the unique identifier of the carousel.

4. A previous session may be resumed by providing ***saved_context*** from a previous ***detach*** operation.

5. Depending on the ***path_name***, this operation returns object references for a service gateway and (optionally) for a first service. If the first service is a composite object, then object references for the parent and child objects will be returned.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.3.

DSM::ServiceGateway::bind - bind object reference to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_bind
              ( DSM_ServiceGateway      gateway1,
              CosNaming_Name            * name1,
              CORBA_Object              obj1,
              CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the object is to be bound
obj1	*(in)* the object to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.5.

DSM::ServiceGateway::bind_context - bind directory to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_bind_context
              ( DSM_ServiceGateway          gateway1,
              CosNaming_Name                * name1,
              CosNaming_NamingContext       dir2,
              CORBA_Environment             * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***).

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.6.

DSM::ServiceGateway::close - close a reference to a service gateway

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_ServiceGateway_close ( DSM_ServiceGateway        gateway1,
                    CORBA_Environment                * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a service gateway is no longer required. The object reference ***obj1*** is deleted, and it is no longer possible to communicate with the service gateway.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::ServiceGateway::destroy - destroy a directory

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_ServiceGateway_destroy ( DSM_ServiceGateway      gateway1,
                                      CORBA_Environment       * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a service gateway, and releases the associated resources.

Notes

1. After this operation, the service gateway no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::ServiceGateway::detach - detach from a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

void     DSM_ServiceGateway_detach
                ( DSM_ServiceGateway        gateway1,
                CORBA_boolean               suspend,
                DSM_UserContext             ** saved_context,
                CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway for the session
suspend	*(in)* whether or not to save the current application context
saved_context	*(out)* saved application context
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_SUSPEND - the application state cannot be saved
(also standard exceptions)

Description

Detach from a service gateway domain, disconnecting from the gateway and from all objects of a session.

Notes

1. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.4.

DSM::ServiceGateway::get - return attribute values bound to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_get
              ( DSM_ServiceGateway      gateway1,
              DSM_PathType              path_type,
              DSM_PathSpec              *path_spec,
              DSM_PathValues            ** path_values,
              CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which attribute values are sought
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(out)* attribute values returned
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Returns attribute values corresponding to the given path specification.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is returned.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be returned, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGateway::Hist - version and time of a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_ServiceGateway_Hist_T ;

DSM_ServiceGateway_Hist_T *  DSM_ServiceGateway__get_Hist
                                ( DSM_ServiceGateway            gateway1,
                                CORBA_Environment               * ev ) ;

void                         DSM_ServiceGateway__set_Hist
                                ( DSM_ServiceGateway            gateway1,
                                DSM_ServiceGateway_Hist_T       * hist1,
                                CORBA_Environment               * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose history is to be returned or modified
hist1	*(in)* the new update time and version of the object
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGateway__get_Hist returns the ***Hist*** attribute of a service gateway.
DSM_ServiceGateway__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a service gateway.

Notes

1. The time is when the service gateway was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_ServiceGateway__get_Hist*** requires ***READER*** privilege. ***DSM_ServiceGateway__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGateway::list - return list of bindings

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_list
              ( DSM_ServiceGateway          gateway1,
              unsigned long                 how_many,
              CosNaming_BindingList         ** binding_list,
              CosNaming_BindingIterator     * binding_iterator,
              CORBA_Environment             * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which bindings are to be listed
how_many	*(in)* the number of bindings to return initially
binding_list	*(out)* the initial list of bindings
binding_iterator	*(out)* the iterator object used to obtain additional bindings
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Returns a list of bindings from a service gateway.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.6.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.3.

DSM::ServiceGateway::Lock - status of service gateway read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_ServiceGateway_Lock_T ;

DSM_ServiceGateway_Lock_T *  DSM_ServiceGateway__get_Lock
                                ( DSM_ServiceGateway        gateway1,
                                CORBA_Environment           * ev ) ;

void                         DSM_ServiceGateway__set_Lock
                                ( DSM_ServiceGateway        gateway1,
                                DSM_ServiceGateway_Lock_T   * lock1,
                                CORBA_Environment           * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGateway__get_Lock returns ***DSM_ServiceGateway_Lock_T***
DSM_ServiceGateway__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a service gateway.

Notes

1. ***DSM_ServiceGateway__get_Lock*** requires ***READER*** privilege. ***DSM_ServiceGateway__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGateway::new_context - create a new directory

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_ServiceGateway_new_context
                ( DSM_ServiceGateway        gateway1,
                DSM_Directory               * new_directory,
                CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway where the new context is to be created
new_directory	*(in)* the new directory
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Creates a new directory on a service gateway.

Notes

1. Requires ***OWNER*** privilege.

2. The new directory is not bound to any context.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.4.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.10.

DSM::ServiceGateway::open - resolve the objects of a path

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_ServiceGateway_open
                ( DSM_ServiceGateway    gateway1,
                DSM_PathType            path_type,
                DSM_PathSpec            * path_spec,
                DSM_ObjRefs             ** resolved_references,
                CORBA_Environment       * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to be used as context for the path specification
path_type	*(in)* the method of path traversal
path_spec	*(in)* the path to be resolved
resolved_references	*(out)* the resolved objects from the path
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

DSM::OPEN_LIMIT - attempt to surpass the maximum allowed number of object references

(also standard exceptions)

Description

Finds the objects associated with the names in a given path.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*directory*>, <*object or directory*>. Each node for which ***process = 1*** is resolved to an object reference in the output list.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to objects within the naming context. The objects (which may be directory objects) are all within the given directory, and each (for which ***process = 1***) is resolved to an object reference for the output list.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The function of this operation is similar to that of ***resolve***. However, ***list*** allows multiple simultaneous resolutions of names.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.13.

DSM::ServiceGateway::Perms - access permission information for a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_ServiceGateway_Perms_T ;

DSM_ServiceGateway_Perms_T *DSM_ServiceGateway__get_Perms
                                ( DSM_ServiceGateway          gateway1,
                                CORBA_Environment             * ev ) ;

void                            DSM_ServiceGateway__set_Perms
                                ( DSM_ServiceGateway          gateway1,
                                DSM_ServiceGateway_Perms_T    * perms1,
                                CORBA_Environment             * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGateway__get_Perms returns ***DSM_ServiceGateway_Perms_T***
DSM_ServiceGateway__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a service gateway.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this service gateway.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGateway::put - bind attribute values to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_put
              ( DSM_ServiceGateway      gateway1,
              DSM_PathType              path_type,
              DSM_PathSpec              *path_spec,
              DSM_PathValues            * path_values,
              CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to which attribute values are to be bound
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(in)* attribute values
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Sets attribute values corresponding to the given path specification.

Notes

1. Requires ***WRITER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is bound.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be bound, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGateway::rebind - rename an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_rebind
              ( DSM_ServiceGateway      gateway1,
              CosNaming_Name            * new_name,
              CORBA_Object              obj1,
              CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the name is to be changed
new_name	*(in)* the new name for the object
obj1	*(in)* the object to be renamed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Renames an object within a given service gateway. This operation replaces an existing name binding in the service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.7.

DSM::ServiceGateway::rebind_context - rename a directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_rebind_context
              ( DSM_ServiceGateway          gateway1,
              CosNaming_Name                * new_name,
              CosNaming_NamingContext       dir2,
              CORBA_Environment             * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the new name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Binds a new name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***). An existing binding for the same name in the service gateway is destroyed.

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.8.

DSM::ServiceGateway::resolve - return object reference for given name

Synopsis - C

```
#include      <dsmcc.h>

CORBA_Object  DSM_ServiceGateway_resolve
                    ( DSM_ServiceGateway      gateway1,
                    CosNaming_Name            * name1,
                    CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to be sought
name1	*(in)* the name for which the binding is to be located
ev	*(in/out)* CORBA environment

Returns

an object reference for the binding in the given context

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Given a name, returns an object reference for an object.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.2.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.4.

DSM::ServiceGateway::Size - size of a directory

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong      DSM_ServiceGateway__get_Size
                        ( DSM_ServiceGateway        gateway1,
                        CORBA_Environment           * ev ) ;
```

Arguments

obj1	*(in)* the service gateway whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a service gateway

Exceptions

(standard)

Description

Returns the size of a service gateway's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGateway::unbind - remove name from an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGateway_unbind
              ( DSM_ServiceGateway      gateway1,
              CosNaming_Name            * name1,
              CORBA_Environment         * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding exists
name1	*(in)* the name for which the binding is to be removed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Remove the binding for a name within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.3.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.9.

DSM::ServiceGatewaySI::attach - attach to a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                           _maximum ;
        unsigned long                           _length ;
        unsigned char                           * _buffer ;
        }
        DSM_Session_ServiceDomain ;

typedef DSM_opaque                              DSM_UserContext ;

typedef struct
        {
        unsigned long                           _maximum ;
        unsigned long                           _length ;
        CORBA_Object                            * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_ServiceGatewaySI_attach
                ( DSM_ServiceGatewaySI          gateway1,
                DSM_Session_ServiceDomain       * server_id,
                CosNaming_Name                  * path_name,
                DSM_UserContext                 * saved_context,
                DSM_ObjRefs                     ** resolved_refs,
                CORBA_Environment               * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway for the session
server_id	*(in)* server identifier (optional)
path_name	*(in)* path name to resolve (optional)
saved_context	*(in)* previous application context (optional)
resolved_refs	*(out)* resolved object references (see Notes, below)
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::OPEN_LIMIT - attempt to surpass the maximum allowable number of active object references

DSM::NO_AUTH - the end-user has not provided the correct authentication in the request

DSM::UNK_USER - the end user is unknown to the service domain
DSM::SERVICE_XFR -a resolve operation was unsuccesful (an alternate service domain location is provided)
DSM::BAD_COMPAT_INFO - an unrecognized compatibility descriptor was provided
DSM::NO_RESUME - the previous application saved state cannot be recovered
DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object
DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path
DSM::INV_NAME - a path name is incorrectly formatted
(also standard exceptions)

Description

Attach to a service gateway domain, establishing a session context.

Notes

1. Requires ***READER*** privilege.

2. Either ***server_id*** or ***path_name*** (or both) must be provided. If both are given, then ***server_id*** specifies the server, and ***path_name*** gives the path to a service gateway and (optionally) to a first service.

3. The ***server_id*** must be in NSAP address format. For an interactive session, it is the globally unique server network address. For a broadcast carousel session, it is the unique identifier of the carousel.

4. A previous session may be resumed by providing ***saved_context*** from a previous ***detach*** operation.

5. Depending on the ***path_name***, this operation returns object references for a service gateway and (optionally) for a first service. If the first service is a composite object, then object references for the parent and child objects will be returned.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.3.

DSM::ServiceGatewaySI::bind - bind object reference to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_bind
              ( DSM_ServiceGatewaySI      gateway1,
              CosNaming_Name              * name1,
              CORBA_Object                obj1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the object is to be bound
obj1	*(in)* the object to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.5.

DSM::ServiceGatewaySI::bind_context - bind directory to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_bind_context
              ( DSM_ServiceGatewaySI      gateway1,
              CosNaming_Name              * name1,
              CosNaming_NamingContext     dir2,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***).

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.6.

DSM::ServiceGatewaySI::close - close a reference to a service gateway

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_ServiceGatewaySI_close ( DSM_ServiceGatewaySI     gateway1,
                    CORBA_Environment                 * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a service gateway is no longer required. The object reference ***gateway1*** is deleted, and it is no longer possible to communicate with the service gateway.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::ServiceGatewaySI::destroy - destroy a directory

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_ServiceGatewaySI_destroy ( DSM_ServiceGatewaySI gateway1,
                                        CORBA_Environment          * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a service gateway, and releases the associated resources.

Notes

1. After this operation, the object no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::ServiceGatewaySI::detach - detach from a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

void    DSM_ServiceGatewaySI_detach
                ( DSM_ServiceGatewaySI      gateway1,
                CORBA_boolean               suspend,
                DSM_UserContext             ** saved_context,
                CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway for the session
suspend	*(in)* whether or not to save the current application context
saved_context	*(out)* saved application context
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_SUSPEND - the application state cannot be saved
(also standard exceptions)

Description

Detach from a service gateway domain, disconnecting from the gateway and from all objects of a session.

Notes

1. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.4.

DSM::ServiceGatewaySI::get - return attribute values bound to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_get
              ( DSM_ServiceGatewaySI      gateway1,
              DSM_PathType                path_type,
              DSM_PathSpec                *path_spec,
              DSM_PathValues              ** path_values,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which attribute values are sought
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(out)* attribute values returned
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Returns attribute values corresponding to the given path specification.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is returned.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be returned, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGatewaySI::Hist - version and time of a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_ServiceGatewaySI_Hist_T ;

DSM_ServiceGatewaySI_Hist_T *DSM_ServiceGatewaySI__get_Hist
                                ( DSM_ServiceGatewaySI          gateway1,
                                CORBA_Environment               * ev ) ;

void                            DSM_ServiceGatewaySI__set_Hist
                                ( CORBA_Object                  gateway1,
                                DSM_ServiceGatewaySI_Hist_T     * hist1,
                                CORBA_Environment               * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose history is to be returned or modified
hist1	*(in)* the new update time and version of the object
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewaySI__get_Hist returns the ***Hist*** attribute of a service gateway. ***DSM_ServiceGatewaySI__set_Hist*** returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a service gateway.

Notes

1. The time is when the service gateway was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_ServiceGatewaySI__get_Hist*** requires ***READER*** privilege. ***DSM_ServiceGatewaySI__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewaySI::list - return list of bindings

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_list
              ( DSM_ServiceGatewaySI      gateway1,
              unsigned long               how_many,
              CosNaming_BindingList       ** binding_list,
              CosNaming_BindingIterator   * binding_iterator,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which bindings are to be listed
how_many	*(in)* the number of bindings to return initially
binding_list	*(out)* the initial list of bindings
binding_iterator	*(out)* the iterator object used to obtain additional bindings
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Returns a list of bindings from a service gateway.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.6.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.3.

DSM::ServiceGatewaySI::Lock - status of service gateway read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_ServiceGatewaySI_Lock_T ;

DSM_ServiceGatewaySI_Lock_T *       DSM_ServiceGatewaySI__get_Lock
                                    ( DSM_ServiceGatewaySI          gateway1,
                                    CORBA_Environment               * ev ) ;

void                                DSM_ServiceGatewaySI__set_Lock
                                    ( DSM_ServiceGatewaySI          gateway1,
                                    DSM_ServiceGatewaySI_Lock_T     * lock1,
                                    CORBA_Environment               * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewaySI__get_Lock returns ***DSM_ServiceGatewaySI_Lock_T***
DSM_ServiceGatewaySI__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a service gateway.

Notes

1. ***DSM_ServiceGatewaySI__get_Lock*** requires ***READER*** privilege. ***DSM_ServiceGatewaySI__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewaySI::new_context - create a new directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_new_context
              ( DSM_ServiceGatewaySI      gateway1,
              DSM_Directory               * new_directory,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway where the new context is to be created
new_directory	*(in)* the new directory
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Creates a new directory on a service gateway.

Notes

1. Requires ***OWNER*** privilege.

2. The new directory is not bound to any context.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.4.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.10.

DSM::ServiceGatewaySI::open - resolve the objects of a path

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_ServiceGatewaySI_open
                ( DSM_ServiceGatewaySI  gateway1,
                DSM_PathType            path_type,
                DSM_PathSpec            * path_spec,
                DSM_ObjRefs             ** resolved_references,
                CORBA_Environment       * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to be used as context for the path specification
path_type	*(in)* the method of path traversal
path_spec	*(in)* the path to be resolved
resolved_references	*(out)* the resolved objects from the path
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

DSM::OPEN_LIMIT - attempt to surpass the maximum allowed number of object references

(also standard exceptions)

Description

Finds the objects associated with the names in a given path.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*directory*>, <*object or directory*>. Each node for which ***process = 1*** is resolved to an object reference in the output list.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to objects within the naming context. The objects (which may be directory objects) are all within the given directory, and each (for which ***process = 1***) is resolved to an object reference for the output list.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The function of this operation is similar to that of ***resolve***. However, ***list*** allows multiple simultaneous resolutions of names.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.13.

DSM::ServiceGatewaySI::Perms - access permission information for a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_ServiceGatewaySI_Perms_T ;

DSM_ServiceGatewaySI_Perms_T *DSM_ServiceGatewaySI__get_Perms
                                        ( DSM_ServiceGatewaySI          gateway1,
                                        CORBA_Environment               * ev ) ;

void                    DSM_ServiceGatewaySI__set_Perms
                                ( DSM_ServiceGatewaySI                  gateway1,
                                DSM_ServiceGatewaySI_Perms_T            * perms1,
                                CORBA_Environment                       * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewaySI__get_Perms returns ***DSM_ServiceGatewaySI_Perms_T***
DSM_ServiceGatewaySI__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a service gateway.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this service gateway.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewaySI::put - bind attribute values to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_put
              ( DSM_ServiceGatewaySI      gateway1,
              DSM_PathType                path_type,
              DSM_PathSpec                *path_spec,
              DSM_PathValues              * path_values,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to which attribute values are to be bound
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(in)* attribute values
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Sets attribute values corresponding to the given path specification.

Notes

1. Requires ***WRITER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is bound.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be bound, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGatewaySI::rebind - rename an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_rebind
              ( DSM_ServiceGatewaySI      gateway1,
              CosNaming_Name              * new_name,
              CORBA_Object                obj1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the name is to be changed
new_name	*(in)* the new name for the object
obj1	*(in)* the object to be renamed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Renames an object within a given service gateway. This operation replaces an existing name binding in the service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.7.

DSM::ServiceGatewaySI::rebind_context - rename a directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_rebind_context
              ( DSM_ServiceGatewaySI        gateway1,
              CosNaming_Name                * new_name,
              CosNaming_NamingContext       dir2,
              CORBA_Environment             * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the new name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Binds a new name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***). An existing binding for the same name in the service gateway is destroyed.

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.8.

DSM::ServiceGatewaySI::resolve - return object reference for given name

Synopsis - C

```
#include      <dsmcc.h>

CORBA_Object  DSM_ServiceGatewaySI_resolve
                    ( DSM_ServiceGatewaySI      gateway1,
                    CosNaming_Name              * name1,
                    CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to be sought
name1	*(in)* the name for which the binding is to be located
ev	*(in/out)* CORBA environment

Returns

an object reference for the binding in the given context

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Given a name, returns an object reference for an object.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.2.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.4.

DSM::ServiceGatewaySI::Size - size of a directory

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong      DSM_ServiceGatewaySI__get_Size
                    ( DSM_ServiceGatewaySI      gateway1,
                    CORBA_Environment           * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a service gateway

Exceptions

(standard)

Description

Returns the size of a service gateway's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewaySI::unbind - remove name from an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewaySI_unbind
              ( DSM_ServiceGatewaySI      gateway1,
              CosNaming_Name              * name1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding exists
name1	*(in)* the name for which the binding is to be removed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Remove the binding for a name within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.3.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.9.

DSM::ServiceGatewayUU::bind - bind object reference to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_bind
              ( DSM_ServiceGatewayUU      gateway1,
              CosNaming_Name              * name1,
              CORBA_Object                obj1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the object is to be bound
obj1	*(in)* the object to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.5.

DSM::ServiceGatewayUU::bind_context - bind directory to name

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_bind_context
              ( DSM_ServiceGatewayUU      gateway1,
              CosNaming_Name              * name1,
              CosNaming_NamingContext     dir2,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
CosNaming::AlreadyBound - an object is already bound to the specified name
(also standard exceptions)

Description

Binds a name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***).

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.6.

DSM::ServiceGatewayUU::close - close a reference to a directory

Synopsis - C

```
#include        <dsmcc.h>

void    DSM_ServiceGatewayUU_close ( DSM_ServiceGatewayUU        gateway1,
                CORBA_Environment               * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a service gateway is no longer required. The object reference ***gateway1*** is deleted, and it is no longer possible to communicate with the service gateway.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::ServiceGatewayUU::destroy - destroy a directory

Synopsis - C

```
#include        <dsmcc.h>

void    DSM_ServiceGatewayUU_destroy ( DSM_ServiceGatewayUU     gateway1,
                                       CORBA_Environment        * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a service gateway, and releases the associated resources.

Notes

1. After this operation, the object no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::ServiceGatewayUU::get - return attribute values bound to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_get
                ( DSM_ServiceGatewayUU      gateway1,
                DSM_PathType                path_type,
                DSM_PathSpec                *path_spec,
                DSM_PathValues              ** path_values,
                CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which attribute values are sought
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(out)* attribute values returned
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Returns attribute values corresponding to the given path specification.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is returned.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be returned, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGatewayUU::Hist - version and time of a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_ServiceGatewayUU_Hist_T ;

DSM_ServiceGatewayUU_Hist_T *       DSM_ServiceGatewayUU__get_Hist
                                    ( DSM_ServiceGatewayUU      gateway1,
                                    CORBA_Environment           * ev ) ;

void                                DSM_ServiceGatewayUU__set_Hist
                                    ( DSM_ServiceGatewayUU      gateway1,
                                    DSM_ServiceGatewayUU_Hist_T * hist1,
                                    CORBA_Environment           * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose history is to be returned or modified
hist1	*(in)* the new update time and version of the object
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewayUU__get_Hist returns the ***Hist*** attribute of a service gateway
DSM_ServiceGatewayUU__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a service gateway.

Notes

1. The time is when the service gateway was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_ServiceGatewayUU__get_Hist*** requires ***READER*** privilege. ***DSM_ServiceGatewayUU__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewayUU::list - return list of bindings

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_list
              ( DSM_ServiceGatewayUU      gateway1,
              unsigned long               how_many,
              CosNaming_BindingList       ** binding_list,
              CosNaming_BindingIterator   * binding_iterator,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway from which bindings are to be listed
how_many	*(in)* the number of bindings to return initially
binding_list	*(out)* the initial list of bindings
binding_iterator	*(out)* the iterator object used to obtain additional bindings
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Returns a list of bindings from a service gateway.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.6.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.3.

DSM::ServiceGatewayUU::Lock - status of service gateway read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean           readLock ;
        CORBA_boolean           writeLock ;
        }
        DSM_ServiceGatewayUU_Lock_T ;

DSM_ServiceGatewayUU_Lock_T *       DSM_ServiceGatewayUU__get_Lock
                                    ( DSM_ServiceGatewayUU      gateway1,
                                    CORBA_Environment           * ev ) ;

void                                DSM_ServiceGatewayUU__set_Lock
                                    ( DSM_ServiceGatewayUU      gateway1,
                                    DSM_ServiceGatewayUU_Lock_T* lock1,
                                    CORBA_Environment           * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewayUU__get_Lock returns ***DSM_ServiceGatewayUU_Lock_T***
DSM_ServiceGatewayUU__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a service gateway.

Notes

1. ***DSM_ServiceGatewayUU__get_Lock*** requires ***READER*** privilege. ***DSM_ServiceGatewayUU__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewayUU::new_context - create a new directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_new_context
              ( DSM_ServiceGatewayUU      gateway1,
              DSM_Directory               * new_directory,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway where the new context is to be created
new_directory	*(in)* the new directory
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

(standard)

Description

Creates a new directory on a service gateway.

Notes

1. Requires ***OWNER*** privilege.

2. The new directory is not bound to any context.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.4.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.10.

DSM::ServiceGatewayUU::open - resolve the objects of a path

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_ServiceGatewayUU_open
                ( DSM_ServiceGatewayUU  gateway1,
                DSM_PathType            path_type,
                DSM_PathSpec            * path_spec,
                DSM_ObjRefs             ** resolved_references,
                CORBA_Environment       * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to be used as context for the path specification
path_type	*(in)* the method of path traversal
path_spec	*(in)* the path to be resolved
resolved_references	*(out)* the resolved objects from the path
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

DSM::OPEN_LIMIT - attempt to surpass the maximum allowed number of object references

(also standard exceptions)

Description

Finds the objects associated with the names in a given path.

Notes

1. Requires ***READER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*directory*>, <*object or directory*>. Each node for which ***process = 1*** is resolved to an object reference in the output list.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to objects within the naming context. The objects (which may be directory objects) are all within the given directory, and each (for which ***process = 1***) is resolved to an object reference for the output list.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The function of this operation is similar to that of ***resolve***. However, ***list*** allows multiple simultaneous resolutions of names.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.13.

DSM::ServiceGatewayUU::Perms - access permission information for a service gateway

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_ServiceGateway_Perms_T ;

DSM_ServiceGatewayUU_Perms_T *DSM_ServiceGatewayUU__get_Perms
                                        ( DSM_ServiceGatewayUU          gateway1,
                                        CORBA_Environment               * ev ) ;

void                    DSM_ServiceGatewayUU__set_Perms
                                ( DSM_ServiceGatewayUU                  gateway1,
                                DSM_ServiceGatewayUU_Perms_T            * perms1,
                                CORBA_Environment               * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_ServiceGatewayUU__get_Perms returns ***DSM_ServiceGatewayUU_Perms_T***
DSM_ServiceGatewayUU__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a service gateway.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this service gateway.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewayUU::put - bind attribute values to a path specification

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_put
              ( DSM_ServiceGatewayUU      gateway1,
              DSM_PathType                path_type,
              DSM_PathSpec                *path_spec,
              DSM_PathValues              * path_values,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway to which attribute values are to be bound
path_type	*(in)* the manner of path traversal
path_spec	*(in)* the path specification
path_values	*(in)* attribute values
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_AUTH - the end user has not provided the correct authentication in the request

DSM::UNK_USER - the Principal (end user) is unknown in the service domain

DSM::SERVICE_XFR - a resolve operation was unsuccessful (an alternate service domain location is provided)

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Sets attribute values corresponding to the given path specification.

Notes

1. Requires ***WRITER*** privilege.

2. If ***path_type*** is ***DSM_PathType_DEPTH***, then the steps of ***path_spec*** correspond to <*directory*>, <*directory*>, ..., <*object*>, <*attribute*>. The last step names the attribute. Only a single attribute value is bound.

3. If ***path_type*** is ***DSM_PathType_BREADTH***, then the steps of ***path_spec*** correspond to <*object*>, <*attribute*>, <*attribute*>, ..., <*attribute*>. Multiple attribute values may be bound, depending on the values of ***process*** in the attribute steps.

4. Each path node is examined sequentially, but not atomically. Other operations may occur between processing individual nodes.

5. If the resolution of any specific node fails, then the entire operation returns the appropriate exception.

6. The kind of attribute (in each path specification step) does not need to be specified. (That is, the value may be NULL.) The identifier is the name of the attribute.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.15.

DSM::ServiceGatewayUU::rebind - rename an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_rebind
              ( DSM_ServiceGatewayUU      gateway1,
              CosNaming_Name              * new_name,
              CORBA_Object                obj1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the name is to be changed
new_name	*(in)* the new name for the object
obj1	*(in)* the object to be renamed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Renames an object within a given service gateway. This operation replaces an existing name binding in the service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.7.

DSM::ServiceGatewayUU::rebind_context - rename a directory

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_rebind_context
              ( DSM_ServiceGatewayUU      gateway1,
              CosNaming_Name              * new_name,
              CosNaming_NamingContext     dir2,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to occur
name1	*(in)* the new name to which the naming context is to be bound
dir2	*(in)* the naming context to be bound to the name
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Binds a new name to an object within a service gateway. (Binds ***dir2*** to ***name1*** within ***gateway1***). An existing binding for the same name in the service gateway is destroyed.

Notes

1. Requires ***MANAGER*** privilege.

2. ***dir2*** may be ***CosNaming::NamingContext***, ***DSM::Directory***, ***DSM::ServiceGateway***, or another object inheriting ***CosNaming::NamingContext***.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.1.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.8.

DSM::ServiceGatewayUU::resolve - return object reference for given name

Synopsis - C

```
#include      <dsmcc.h>

CORBA_Object  DSM_ServiceGatewayUU_resolve
                    ( DSM_ServiceGatewayUU      gateway1,
                    CosNaming_Name              * name1,
                    CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding is to be sought
name1	*(in)* the name for which the binding is to be located
ev	*(in/out)* CORBA environment

Returns

an object reference for the binding in the given context

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Given a name, returns an object reference for an object.

Notes

1. Requires ***READER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.2.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.4.

DSM::ServiceGatewayUU::Size - size of a directory

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong      DSM_ServiceGatewayUU__get_Size
                        ( DSM_ServiceGatewayUU      gateway1,
                        CORBA_Environment           * ev ) ;
```

Arguments

gateway1	*(in)* the service gateway whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a service gateway

Exceptions

(standard)

Description

Returns the size of a service gateway's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::ServiceGatewayUU::unbind - remove name from an object

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_ServiceGatewayUU_unbind
              ( DSM_ServiceGatewayUU      gateway1,
              CosNaming_Name              * name1,
              CORBA_Environment           * ev1 ) ;
```

Arguments

gateway1	*(in)* the service gateway in which the binding exists
name1	*(in)* the name for which the binding is to be removed
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

CosNaming::NotFound - the name does not identify a binding
CosNaming::CannotProceed - the implementation has given up for some reason
CosNaming::InvalidName - the name is invalid
(also standard exceptions)

Description

Remove the binding for a name within a service gateway.

Notes

1. Requires ***MANAGER*** privilege.

Reference

Object Management Group, *CORBAservices: Common Object Services Specification*, July 1996, 3.2.3.

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.5.9.

DSM::Session::attach - attach to a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        unsigned char                   * _buffer ;
        }
        DSM_Session_ServiceDomain ;

typedef DSM_opaque                      DSM_UserContext ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Object                    * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_Session_attach
                ( DSM_Session           session1,
                DSM_Session_ServiceDomain * server_id,
                CosNaming_Name          * path_name,
                DSM_UserContext         * saved_context,
                DSM_ObjRefs             ** resolved_refs,
                CORBA_Environment       * ev1 ) ;
```

Arguments

session1	*(in)* the local session object
server_id	*(in)* server identifier (optional)
path_name	*(in)* path name to resolve (optional)
saved_context	*(in)* previous application context (optional)
resolved_refs	*(out)* resolved object references (see Notes, below)
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::OPEN_LIMIT - attempt to surpass the maximum allowable number of active object references

DSM::NO_AUTH - the end-user has not provided the correct authentication in the request

DSM::UNK_USER - the end user is unknown to the service domain
DSM::SERVICE_XFR -a resolve operation was unsuccesful (an alternate service domain location is provided)
DSM::BAD_COMPAT_INFO - an unrecognized compatibility descriptor was provided
DSM::NO_RESUME - the previous application saved state cannot be recovered
DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object
DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path
DSM::INV_NAME - a path name is incorrectly formatted
(also standard exceptions)

Description

Attach to a service gateway domain, establishing a session context.

Notes

1. Requires ***READER*** privilege.

2. Either ***server_id*** or ***path_name*** (or both) must be provided. If both are given, then ***server_id*** specifies the server, and ***path_name*** gives the path to a service gateway and (optionally) to a first service.

3. The ***server_id*** must be in NSAP address format. For an interactive session, it is the globally unique server network address. For a broadcast carousel session, it is the unique identifier of the carousel.

4. A previous session may be resumed by providing ***saved_context*** from a previous ***detach*** operation.

5. Depending on the ***path_name***, this operation returns object references for a service gateway and (optionally) for a first service. If the first service is a composite object, then object references for the parent and child objects will be returned.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.3.

DSM::Session::detach - detach from a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

void     DSM_Session_detach
                ( DSM_Session               session1,
                CORBA_boolean               suspend,
                DSM_UserContext             ** saved_context,
                CORBA_Environment           * ev1 ) ;
```

Arguments

session1	*(in)* the local session object
suspend	*(in)* whether or not to save the current application context
saved_context	*(out)* saved application context
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_SUSPEND - the application state cannot be saved
(also standard exceptions)

Description

Detach from a service gateway domain, disconnecting from the gateway and from all objects of a session.

Notes

1. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.6.4.

DSM::SessionSI::attach - attach to a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

typedef struct
        {
        unsigned long                       _maximum ;
        unsigned long                       _length ;
        CORBA_Object                        * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_SessionSI_attach
                ( DSM_SessionSI             session1,
                DSM_InfoRequest             * download_info_request,
                CosNaming_Name              * path_name,
                DSM_UserContext             * saved_context,
                DSM_InfoResponse            ** download_info_response,
                DSM_ObjRefs                 ** resolved_refs,
                CORBA_Environment           * ev1 ) ;
```

Arguments

session1	*(in)* the session object
download_info_request	*(in)* download information request
path_name	*(in)* path name to resolve
saved_context	*(in)* previous application context (optional)
download_info_response	*(out)* download information response
resolved_refs	*(out)* resolved object references (see Notes, below)
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::OPEN_LIMIT - attempt to surpass the maximum allowable number of active object references

DSM::NO_AUTH - the end-user has not provided the correct authentication in the request

DSM::UNK_USER - the end user is unknown to the service domain

DSM::SERVICE_XFR -a resolve operation was unsuccesful (an alternate service domain location is provided)

DSM::BAD_COMPAT_INFO - an unrecognized compatibility descriptor was provided

DSM::NO_RESUME - the previous application saved state cannot be recovered

DSM::NOT_FOUND - either (1) a logical path name does not exist; (2) an intermediate node is not a directory; or (3) an intermediate node could not resolve the object

DSM::CANNOT_PROCEED - a directory did not have permission to resolve a node in a logical path

DSM::INV_NAME - a path name is incorrectly formatted

(also standard exceptions)

Description

Attach to a service gateway domain, establishing a session context.

Notes

1. Requires ***READER*** privilege.

2. A previous session may be resumed by providing ***saved_context*** from a previous ***detach*** operation.

3. Depending on the ***path_name***, this operation returns object references for a service gateway and (optionally) for a first service. If the first service is a composite object, then object references for the parent and child objects will be returned.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.3.

DSM::SessionSI::detach - detach from a service gateway domain

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

void     DSM_SessionSI_detach
                ( DSM_SessionSI             session1,
                CORBA_boolean               suspend,
                DSM_UserContext             ** saved_context,
                CORBA_Environment           * ev1 ) ;
```

Arguments

session1	*(in)* the session object
suspend	*(in)* whether or not to save the current application context
saved_context	*(out)* saved application context
ev	*(in/out)* CORBA environment

Returns

void

Exceptions

DSM::NO_SUSPEND - the application state cannot be saved
(also standard exceptions)

Description

Detach from a service gateway domain, disconnecting from the gateway and from all objects of a session.

Notes

1. Requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.3.

DSM::SessionUU::attach - define *uuData* for session establishment

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_SessionUU_attach
              ( DSM_SessionUU              user_network_session,
              DSM_opaque                   * download_info_request,
              DSM_CosNaming_Name           * path_name,
              DSM_UserContext              * saved_context,
              DSM_Principal                * principal,
              DSM_IOP_ServiceContextList   * in_service_context_list,
              DSM_opaque                   ** download_info_response,
              DSM_ConnBinder               ** download_taps,
              DSM_ObjRefs                  ** resolved_refs,
              DSM_IOP_ServiceContextList   ** out_service_context_list,
              CORBA_Environment            * ev ) ;
```

Arguments

user_network_session	*(in)* the session for which the uuData is to be defined
download_info_request	*(in)* download info request
path_name	*(in)* path to desired service
saved_context	*(in)* previous application user context
principal	*(in)* identification of end user
in_service_context_list	*(in)* information for service context list
download_info_response	*(out)* download info response
download_taps	*(out)* the download taps to be used for communication with the resolved service object
resolved_refs	*(out)* objects resolved
out_service_context_list	*(out)* returned service context list information
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::NO_AUTH: the end user has not provided the correct authentication in the request
DSM::BAD_COMPAT_INFO: there was an unrecognized ***CompatibilityDescriptor*** in ***download_info_request***
DSM::UNK_USER: the principal end user is unknown in the service domain
DSM::SERVICE_XFR: a resolve operation was unsuccessful (an alternative service domain location is provided)
DSM::NO_RESUME: the previous application state (in ***saved_context***) cannot be recovered
DSM::OPEN_LIMIT: the number of active object references is the maximum allowed

DSM::NOT_FOUND: either (1) a logical path name does not exist, (2) an intermediate node is not a directory, or (3) an object could not be resolved
DSM::CANNOT_PROCEED: a directory did not have permission to resolve a node in a logical path
DSM::INV_NAME: a path name is incorrectly formatted

(also standard exceptions)

Description

Use to specify the paramters to be included in the ***uuData*** fields of the User-Network Session Establishment messages. The input parameters are placed in the U-N ***ClientSessionSetupRequest***, and the output parameters are taken from the U-N ***ClientSessionSetupResponse***.

Notes

1. This procedure is used from within the User-User library, and is not normally used by an application. It is not available on all platforms, so it is not completely portable.

2. If the request is for a download session, then ***download_info_request*** and ***download_info_response*** are significant. (Otherwise, they are zero.)

3. The ***path_name*** has two steps: the path to an object of class ***DSM::ServiceGatewayUU***, and (possibly) the name of a first service. A default service may be specified in the second step as a ***NULL*** string. The resolved object reference(s) are returned in ***resolved_refs***.

4. The taps for initial download of the resolved object are returned in ***download_taps***., which is a sequence of ***DSM::Tap***. If no initial download is required, then this sequence is empty.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.1.

DSM::SessionUU::detach - define *uuData* for session teardown

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_SessionUU_detach
              ( DSM_SessionUU              user_network_session,
              CORBA_boolean                suspend,
              DSM_Principal                * principal,
              DSM_IOP_ServiceContextList   * in_service_context_list,
              DSM_IOP_ServiceContextList   ** out_service_context_list,
              DSM_UserContext              * saved_context,
              CORBA_Environment            * ev ) ;
```

Arguments

user_network_session	*(in)* the session for which the uuData is to be defined
suspend	*(in)* whether or not to save the application context
principal	*(in)* the end user making the request
in_service_context_list	*(in)* information for service context list
out_service_context_list	*(out)* returned service context list information
saved_context	*(out)* the saved application context
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::NO_SUSPEND - the application context cannot be saved
(also standard exceptions)

Description

Use to specify the paramters to be included in the ***uuData*** fields of the User-Network Session Teardown messages.

Notes

1. This procedure is used from within the User-User library, and is not normally used by an application. It is not available on all platforms, so it is not completely portable.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.6.6.1.

DSM::State::resume - resume a service from a previous application state

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

typedef struct
        {
        unsigned long                       _maximum ;
        unsigned long                       _length ;
        CORBA_Object                        * _buffer ;
        }
        DSM_ObjRefs ;

void    DSM_State_resume
                ( CORBA_Object              obj1,
                DSM_UserContext             * saved_context,
                DSM_ObjRefs                 ** restored_references,
                CORBA_Environment           * ev ) ;
```

Arguments

obj1	*(in)* an object whose state was saved previously
saved_context	*(in)* the saved state
restored_references	*(out)* restored object references
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::NO_RESUME - unable to restore the object's context
(also standard exceptions)

Description

Restore an object's saved context.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.5.3.

DSM::State::suspend - suspend application state for a service

Synopsis - C

```
#include        <dsmcc.h>

typedef DSM_opaque                          DSM_UserContext ;

void    DSM_State_suspend
                ( CORBA_Object              obj1,
                CORBA_boolean               release,
                DSM_UserContext             ** saved_context,
                CORBA_Environment           * ev ) ;
```

Arguments

obj1	*(in)* an object whose state is to be saved
release	*(in)* indication if resumption is expected soon
saved_context	*(out)* the saved state
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::NO_SUSPEND - unable to save object's state
(also standard exceptions)

Description

Save an object's state for resumption at a later time.

Notes

1. This operation requires ***READER*** privilege.

2. The ***release*** parameter is a hint to the underlying transport for use in resource allocation.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.2.5.2.

DSM::Stream - the stream interface

An object with a ***DSM::Stream*** interface can transport audio, video, or other program content to a client. The operations emulate VCR-like controls for starting, stopping, fast-forward, and other control mechanisms.

The Stream State Machine: Simplified Version

The behaviour of a ***DSM::Stream*** object is described as a state machine. Transitions among states occur as a result of:

- conditions which arise during search and transport of the stream content
- operations invoked on the ***DSM::Stream*** object by a client.

The state variables are:

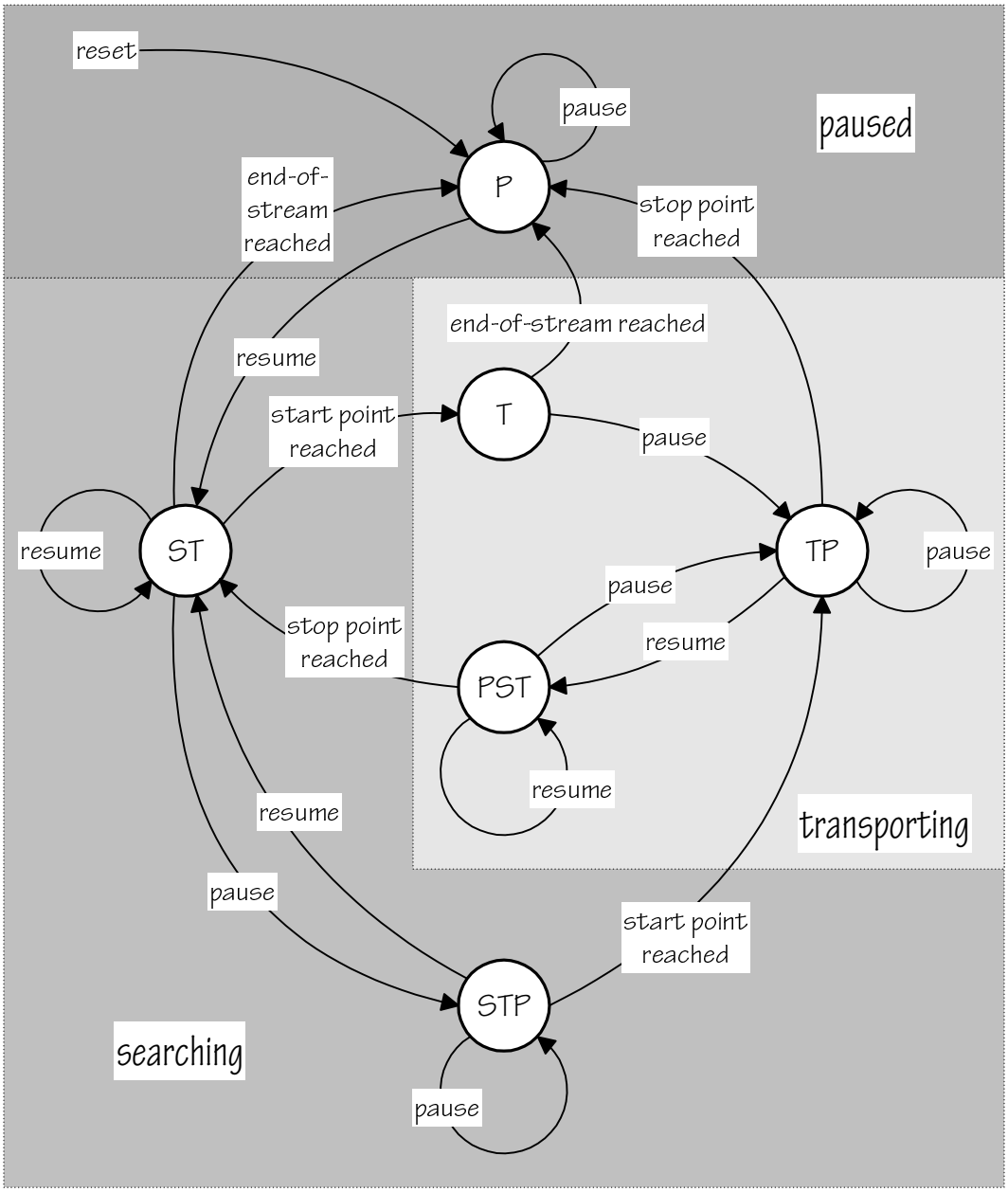
- the current state
- the current position within the stream
- the scale or rate of playback
- a "start" position within the stream (***startNPT***)
- a "stop" position within the stream (***stopNPT***)

The states themselves may be grouped into three categories, based on the transport activity which occurs during each state. The server is either *paused*, *transporting*, or *searching*, depending on the state.

Paused States

- Open (O). This is the initial state which exists when the ***DSM::Stream*** object is resolved. The stream is inactive. The current position is the beginning of the stream.
- Pause (P). During this state, the stream is inactive.
- End of Stream (EOS). This is the same as Pause, except that the current position is the end of the stream.

These three paused states undergo the same state transitions, so they are combined together in the simplified diagram below.



TRANSPORTING STATES

- TRANSPORT (T). The server is transporting, and will continue until end of stream is reached. No stop position within the stream is defined.

- TRANSPORT PAUSE (TP). A stop position is defined. The server will continue transporting until the stop position is reached.

- PAUSE SEARCH TRANSPORT (PST). Both start and stop positions are defined. The server will continue transporting until the stop position is reached, then it will enter the SEARCH TRANSPORT state to search for the start position.

SEARCHING STATES

- SEARCH TRANSPORT (ST). The server is seeking to a defined start position. When it is reached, it will enter the TRANSPORT state and begin transporting until end of stream is reached.

- SEARCH TRANSPORT PAUSE (STP). Both start and stop positions are defined. The server seeks to the start position, then enters TRANSPORT PAUSE state to begin transporting until the stop position is reached.

THE STREAM STATE MACHINE: DETAILED VERSION

The state transition diagram above is simplified by three factors:

1. the elimination of ***play*** and ***jump*** operations

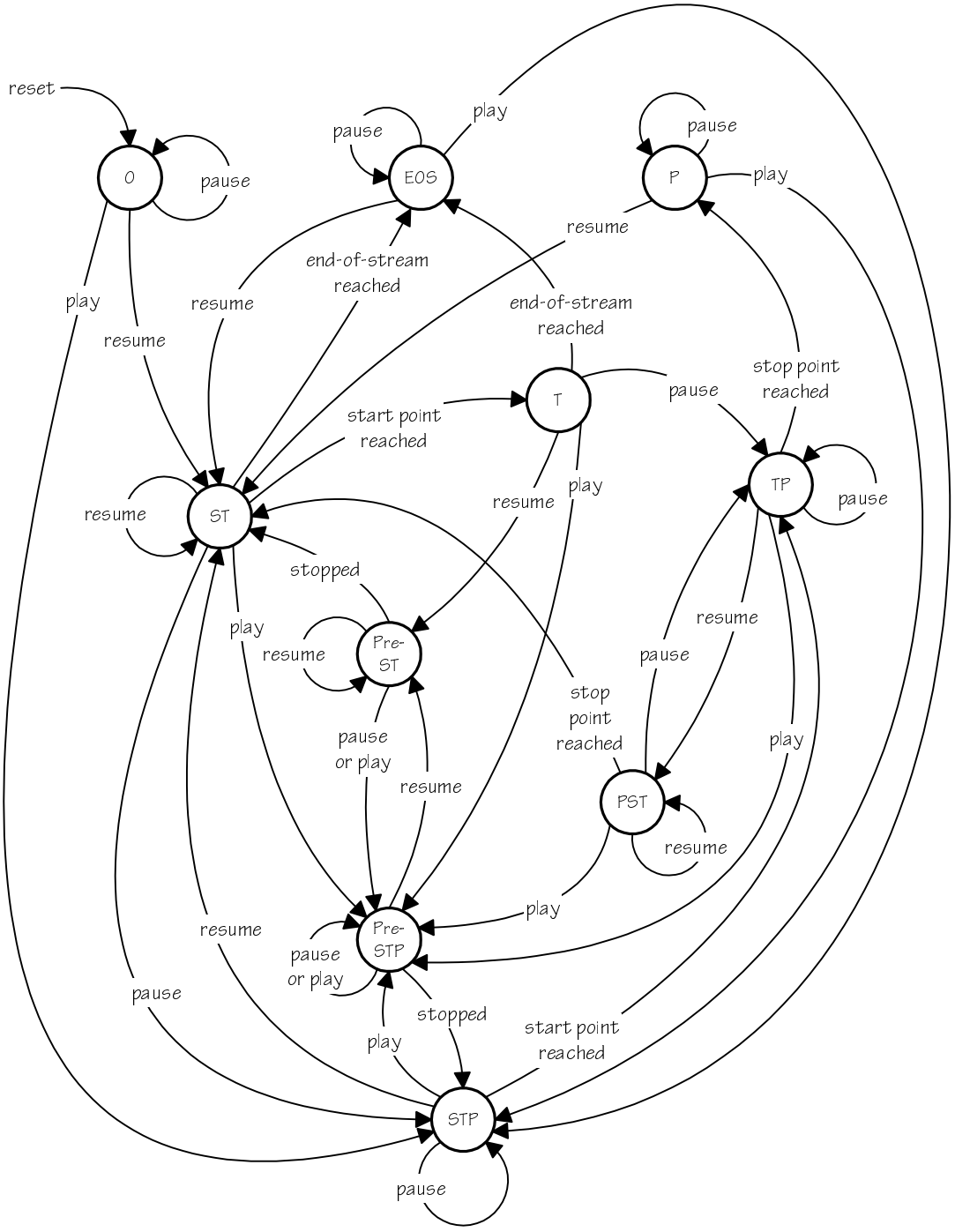
2. the combination of the three pause states (PAUSE, OPEN, and END OF STREAM) into a single PAUSE state

3. the elimination of two transient states, PRE-SEARCH TRANSPORT (PRE-ST) and PRE-SEARCH TRANSPORT PAUSE (PRE-STP)

The two transient states define the behaviour of the state machine when a transition is made from another state to SEARCH TRANSPORT (ST) or to SEARCH TRANSPORT PAUSE (STP) state. During such a transition, the server must stop the current search or transport operation before beginning the new search operation. The transient states (PRE-ST and PRE-STP) are defined as the conditions where the previous operation is being halted, but the next operation is not yet begun.

The following more complete state transition diagram for a ***DSM::Stream*** object includes the ***play*** operation. However, the ***jump*** operation, which is identical to ***pause*** followed by ***resume***, is omitted for readability.

The following diagram also includes the two transient states, and separates the three pause states. Note that a ***reset*** operation can occur from any state, and will return the state machine to the OPEN state.



Normal Play Time and Rate of Play

The stream state machine addresses points in the stream using *Normal Play Time* (NPT) coordinates. NPT is expressed in seconds and microseconds, and is expressed independently of the underlying content frame rate or other factors. The NPT coordinate of the beginning of a stream is defined as zero.

A special value of NPT is 0x80000000, or "negative infinity". This represents "now" as perceived by the server. For example, if the server is told to resume play at 0x80000000, it will resume play at the current position.

While the stream is being transported, the rate of play is determined by the current value of the *scale* parameter. The scale is a fraction (numerator and denominator), and may be negative or positive. A magnitude greater than one indicates an accelerated rate of play, and a magnitude less than one indicates slow motion play.

Negative scale values are used to request reverse play.

Not all implementations will implement all possible rates of play, nor will they all be able to address the stream with the same temporal resolution. The server will use its best effort to satisfy the request.

DSM::Stream::close - close a reference to a stream

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_Stream_close ( DSM_Stream        stream1,
                            CORBA_Environment * ev ) ;
```

Arguments

stream1	*(in)* the stream whose reference is no longer required
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Indicate that access to a stream is no longer required. The object reference ***stream1*** is deleted, and it is no longer possible to communicate with the stream.

Notes

1. Object references are resources. A system may have a limit to the maximum number of open object reference. (An attempt to surpass the limit will result in a ***DSM::OPEN_LIMIT*** exception.) This operation may be used to reduce the number of open references, thus avoiding the exception. It use is prudent in standard practice, but not required.

2. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.2.

DSM::Stream::destroy - destroy a stream

Synopsis - C

```
#include        <dsmcc.h>

void     DSM_Stream_destroy ( DSM_Stream       stream1,
                              CORBA_Environment  * ev ) ;
```

Arguments

stream1	*(in)* the stream to be destroyed
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Destroys a stream, and releases the associated resources.

Notes

1. After this operation, the stream no longer exists, and the object reference is no longer valid.

2. This operation requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.1.3.

DSM::Stream::Hist - version and time of a stream

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                    aMajor ;
        char                    aMinor ;
        }
        DSM_Version ;

typedef struct
        {
        long            tm_sec ;        /* seconds, 0..59 */
        long            tm_min ;        /* minutes, 0..59 */
        long            tm_hour ;       /* hours, 0..23 */
        long            tm_mday ;       /* day of month, 1..31 */
        long            tm_mon ;        /* months since Jan, 0..11 */
        long            tm_year ;       /* years since 1900 */
        long            tm_wday ;       /* days since Sunday, 0..6 */
        long            tm_yday ;       /* days since Jan 1, 0..365 */
        long            tm_isdst ;      /* daylight savings time indicator */
        }
        DSM_DateTime ;

typedef struct
        {
        DSM_Version             aVersion ;
        DSM_DateTime            aDateTime ;
        }
        DSM_Stream_Hist_T ;

DSM_Stream_Hist_T *   DSM_Stream__get_Hist
                                ( DSM_Stream                stream1,
                                CORBA_Environment           * ev ) ;

void                  DSM_Stream__set_Hist
                                ( DSM_Stream                stream1,
                                DSM_Stream_Hist_T           * hist1,
                                CORBA_Environment           * ev ) ;
```

Arguments

stream1	*(in)* the stream whose history is to be returned or modified
hist1	*(in)* the new update time and version of the object
ev	*(in/out)* CORBA environment

Returns

DSM_Stream__get_Hist returns the ***Hist*** attribute of a stream.
DSM_Stream__set_Hist returns void.

Exceptions

(standard)

Description

Returns or sets the version and time of a stream.

Notes

1. The time is when the stream was created or last updated. It is specified in GMT.

2. The ***DSM_DateTime*** structure was modified from the ANSI C standard. It is not, in general, identical to the C structure ***tm***.

3. The field ***tm_isdst*** of ***DSM_DateTime*** indicates the use of Daylight Savings Time (summer time). However, GMT is not defined for GMT, so the interpretation is not clear. The value of the field is defined as follows:

tm_isdst > 0	daylight savings time in effect
tm_isdst = 0	daylight savings time not in effect
tm_isdst < 0	information not available

4. ***DSM_Stream__get_Hist*** requires ***READER*** privilege. ***DSM_Stream__set_Hist*** requires ***BROKER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Stream::info - stream identification and characteristics

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        char                        * aDescription ;
        DSM_AppNPT                  duration ;
        CORBA_boolean               audio ;
        CORBA_boolean               video ;
        CORBA_boolean               data ;
        }
        DSM_Stream_Info_T ;

DSM_Stream_Info_T *   DSM_Stream__get_Info ( DSM_Stream          stream1,
                                             CORBA_Environment  * ev ) ;

void                  DSM_Stream__set_Info ( DSM_Stream          stream1,
                                             DSM_Stream_Info_T  * info1,
                                             CORBA_Environment  * ev ) ;
```

Arguments

stream1	*(in)* the stream whose information is to be returned or set
info1	*(in)* the new value of the information
ev	*(in/out)* CORBA environment

Returns

DSM_Stream__get_Info returns ***DSM_Stream_Info_T***
DSM_Stream__set_Info returns void

Exceptions

(standard)

Description

Returns or sets the information attribute of a stream.

Notes

1. ***DSM_Stream__get_Lock*** requires ***READER*** privilege. ***DSM_Stream__set_Lock*** requires ***OWNER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.1.

DSM::Stream::jump - when stream reaches stop NPT, resume at start NPT

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Stream_jump
              ( DSM_Stream                stream1,
              DSM_AppNPT                  * start,
              DSM_AppNPT                  * stop,
              DSM_Scale                   * scale,
              CORBA_Environment           * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
start	*(in)* the starting time
stop	*(in)* the stopping time
scale	*(in)* the rate and direction of play
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::BAD_SCALE - an invalid scale value was given
DSM::BAD_START - the indicated start time does not exist
DSM::BAD_STOP - the indicated stop time does not exist
DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream
(also standard exceptions)

Description

When the stream reaches ***stop***, resume play at ***start*** with rate and direction ***scale***.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.9.

DSM::Stream::Lock - status of stream read and write locks

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        CORBA_boolean        readLock ;
        CORBA_boolean        writeLock ;
        }
        DSM_Stream_Lock_T ;

DSM_Stream_Lock_T *  DSM_Stream__get_Lock ( DSM_Stream          stream1,
                                           CORBA_Environment   * ev ) ;

void                 DSM_Stream__set_Lock ( DSM_Stream          stream1,
                                           DSM_Access_Lock_T   * lock1,
                                           CORBA_Environment   * ev ) ;
```

Arguments

stream1	*(in)* the stream whose locks are to be returned or set
lock1	*(in)* the new value of the locks
ev	*(in/out)* CORBA environment

Returns

DSM_Stream__get_Lock returns ***DSM_Stream_Lock_T***
DSM_Stream__set_Lock returns void

Exceptions

(standard)

Description

Returns or sets the lock attribute of a stream.

Notes

1. ***DSM_Stream__get_Lock*** requires ***READER*** privilege. ***DSM_Stream__set_Lock*** requires ***WRITER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Stream::pause - stop sending stream when NPT is reached

Synopsis - C

```
#include      <dsmcc.h>

void     DSM_Stream_pause
               ( DSM_Stream                stream1,
               DSM_AppNPT                  * stop,
               CORBA_Environment           * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
stop	*(in)* the stopping time
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::BAD_STOP - the indicated stop time does not exist
DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream
(also standard exceptions)

Description

When the stream reaches ***stop***, halt play.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.5.

DSM::Stream::Perms - access permission information for a stream

Synopsis - C

```
#include        <dsmcc.h>

typedef struct
        {
        unsigned short          managerPerm ;
        unsigned short          brokerPerm ;
        DSM_u_longlong          writerPerm ;
        DSM_u_longlong          readerPerm ;
        DSM_opaque              owner ;
        char                    * aPassword ;
        DSM_opaque              authData ;
        CORBA_boolean           allSecure ;
        }
        DSM_Stream_Perms_T ;

DSM_Stream_Perms_T *    DSM_Stream__get_Perms
                                ( DSM_Stream                stream1,
                                CORBA_Environment           * ev ) ;

void                    DSM_Access__set_Perms
                                ( DSM_Stream                stream1,
                                DSM_Stream_Perms_T          * perms1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

obj1	*(in)* the stream whose permissions are to be set or returned
perms1	*(in)* the new permissions
ev	*(in/out)* CORBA environment

Returns

DSM_Stream__get_Perms returns ***DSM_Stream_Perms_T***
DSM_Stream__set_Perms returns void

Exceptions

(standard)

Description

Returns or sets the permission attribute of a stream.

Notes

1. Both of these operations require ***OWNER*** privilege.

2. ***managerPerm***, ***brokerPerm***, ***writerPerm***, and ***readerPerm*** are bit masks specifying which manager, broker, writer, and reader groups have access to the object.

3. The ***owner*** field is equivalent to CORBA's ***Principal***.

4. If ***aPassword*** is non-null, then a password must be provided for access. If ***authData*** is non-null, then an implementation-dependent encryption challenge must be met before access is permitted. Refer to the discussion of security and authentication in the notes on implementation and use for more information.

5. If ***allSecure*** is true, then all lower communication layers must use encryption when transferring any method parameters for this stream.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Stream::play - play stream from start NPT until stop NPT

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Stream_play
              ( DSM_Stream                stream1,
              DSM_AppNPT                  * start,
              DSM_AppNPT                  * stop,
              DSM_Scale                   * scale,
              CORBA_Environment           * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
start	*(in)* the starting time
stop	*(in)* the stopping time
scale	*(in)* the rate and direction of play
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::BAD_SCALE - an invalid scale value was given
DSM::BAD_START - the indicated start time does not exist
DSM::BAD_STOP - the indicated stop time does not exist
DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream
(also standard exceptions)

Description

Begin playing at ***start*** with rate and direction ***scale***, halting when ***stop*** is reached.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.10.

DSM::Stream::reset - reset a stream state machine

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Stream_reset
              ( DSM_Stream                    stream1,
              CORBA_Environment               * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Reset the stream state machine.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.8.

DSM::Stream::resume - start sending stream at NPT

Synopsis - C

```
#include      <dsmcc.h>

void    DSM_Stream_jump
              ( DSM_Stream                stream1,
              DSM_AppNPT                  * start,
              DSM_Scale                   * scale,
              CORBA_Environment           * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
start	*(in)* the starting time
scale	*(in)* the rate and direction of play
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::BAD_SCALE - an invalid scale value was given
DSM::BAD_START - the indicated start time does not exist
DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream
(also standard exceptions)

Description

Resume play at ***start*** with rate and direction ***scale***.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.6.

DSM::Stream::Size - size of a stream

Synopsis - C

```
#include        <dsmcc.h>

DSM_u_longlong      DSM_Stream__get_Size ( DSM_Stream          stream1,
                                           CORBA_Environment       * ev ) ;
```

Arguments

stream1	*(in)* the stream whose size is sought
ev	*(in/out)* CORBA environment

Returns

the size (in octets) of the attributes of a stream

Exceptions

(standard)

Description

Returns the size of a stream's attributes.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.2.

DSM::Stream::status - obtain status of a stream

Synopsis - C

```
#include      <dsmcc.h>

typedef unsigned long                    DSM_Stream_Mode ;

typedef struct
        {
        DSM_AppNPT                       rPosition ;
        DSM_Scale                        rScale ;
        DSM_Stream_Mode                  aMode ;
        }
        DSM_Stream_Stat ;

void    DSM_Stream_status
                ( DSM_Stream                    stream1,
                DSM_Stream_Stat                 * estimated,
                DSM_Stream_Stat                 ** actual,
                CORBA_Environment               * ev ) ;
```

Arguments

stream1	*(in)* the stream being played
estimated	*(in)* estimated status
actual	*(out)* actual status
ev	*(in/out)* CORBA environment

Returns

(void)

Exceptions

DSM::MPEG_DELIVERY - the server was unable to deliver the object over an MPEG stream

(also standard exceptions)

Description

Given an (optional) estimate of current stream status, this operation returns the actual value.

Notes

1. This operation requires ***READER*** privilege.

Reference

ISO/IEC 13818-6, *Digital Storage Media Command & Control*, July 1996, 5.5.1.3.7.

DSM::View::execute - execute SQL write statement

(to be supplied)

DSM::View::query - execute SQL select statement, return initial results

(to be supplied)

DSM::View::read - read additional results in context of SQL query

(to be supplied)

es_rsrc_atm_pvc_info - get ATM PVC connection information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_atm_pvc_info
                    ( CORBA_Object          atm_pvc_connection,
                    es_atm_address_ t       ** atm_address,
                    unsigned short          ** virtual_channel_id,
                    unsigned short          ** virtual_path_id,
                    CORBA_Environment       * ev ) ;
```

Arguments

atm_pvc_connection	*(in)* ATM PVC connection resource object
atm_address	*(out)* the ATM address for the connection
virtual_channel_id	*(out)* the virtual channel id (VCI) for the connection
virtual_path_id	*(out)* the virtual path id (VPI) for the connection
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about an ATM PVC connection resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_atm_vc_info - get ATM VC connection information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_atm_vc_info
                    ( CORBA_Object          atm_vc_connection,
                    unsigned short          ** virtual_channel_id,
                    unsigned short          ** virtual_path_id,
                    CORBA_Environment       * ev ) ;
```

Arguments

atm_pvc_connection	*(in)* ATM PVC connection resource object
virtual_channel_id	*(out)* the virtual channel id (VCI) for the connection
virtual_path_id	*(out)* the virtual path id (VPI) for the connection
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about an ATM VC connection resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_bandwidth_down_info - get downstream transport bandwidth information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_bandwidth_down_info
                    ( CORBA_Object          downstream_bandwidth,
                    unsigned long           * bits_per_second,
                    unsigned long           ** transport_id,
                    CORBA_Environment       * ev ) ;
```

Arguments

downstream_bandwidth	*(in)* the downstream transport bandwidth resource object
bits_per_second	*(out)* data rate allocated to the session
transport_id	*(out)* transport stream identifier (assigned by network)
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about a downstream transport bandwidth resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_bandwidth_up_info - get upstream transport bandwidth information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_bandwidth_up_info
                        ( CORBA_Object          upstream_bandwidth,
                        unsigned long           * bits_per_second,
                        unsigned long           ** transport_id,
                        CORBA_Environment       * ev ) ;
```

Arguments

upstream_bandwidth	*(in)* the upstream transport bandwidth resource object
bits_per_second	*(out)* data rate allocated to the session
transport_id	*(out)* transport stream identifier (assigned by network)
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about an upstream transport bandwidth resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_cfs_info - get continuous feed session information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_cfs_info
                    ( CORBA_Object          cont_feed_session,
                    es_session_id_t         ** session_id,
                    sequence_of_objects     ** resource_list,
                    CORBA_Environment       * ev ) ;
```

Arguments

cont_feed_session	*(in)* continuous feed session resource object
session_id	*(out)* session identifier
resource_list	*(out)* resources used in session
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about a continuous feed session resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_internet_info - get internet connection information

Synopsis - C

```
#include <spain.h>

void     es_rsrc_internet_info
                        ( CORBA_Object          internet_connection,
                        unsigned long           * source_ip_address,
                        unsigned short          * source_port_number,
                        unsigned long           * destination_ip_address,
                        unsigned short          * destination_port_number,
                        unsigned short          * protocol_type,
                        CORBA_Environment       * ev ) ;
```

Arguments

internet_connection	*(in)* internet connection resource object
source_ip_address	*(out)* the IP address of the device sending the messages
source_port_number	*(out)* the port from which data will be sent
destination_ip_address	*(out)* the IP address of the device receiving the messages
destination_port_number	*(out)* the port to which data will be sent
protocol_type	*(out)* the protocol being carried over the IP stream: ***ES_PROTOCOL_TCP_IP*** - TCP/IP ***ES_PROTOCOL_UDP_IP*** - UDP/IP ***ES_PROTOCOL_USER_DEFINED*** - user defined protocol
virtual_path_id	*(out)* the virtual path id (VPI) for the connection
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about an internet connection resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_list - get resources associated with an object

Synopsis - C

```
#include <spain.h>

void     es_rsrc_list
                        ( CORBA_Object          obj1,
                        sequence_of_objects     ** resource_list,
                        CORBA_Environment       * ev ) ;
```

Arguments

obj1	*(in)* a stream, file, download, or other DSM-CC object
resource_list	*(out)* resource objects used by ***obj1***
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns the resource objects associated with a DSM-CC object.

Notes

1. Once ***resource_list*** is obtained, the members of the sequence can be interrogated for more specific information.

Reference

(proprietary definition)

es_rsrc_mpeg_pgm_info - get MPEG program information

Synopsis - C

```
#include <spain.h>

typedef struct
        {
        unsigned short          packet_id ;
        unsigned short          stream_type ;
        es_association_tag_t    association_tag ;
        }
        es_stream_info_t ;

typedef struct
        {
        unsigned long           _maximum ;
        unsigned long           _length ;
        es_stream_info_t        * _buffer ;
        }
        es_stream_table_t ;

void    es_rsrc_mpeg_pgm _info
                        ( CORBA_Object          mpeg_program,
                        unsigned short          * program_number,
                        unsigned short          * map_table_pid,
                        unsigned short          * conditional_access_pid,
                        es_stream_table_t       ** stream_table,
                        short                   * program_clock_ref_pid,
                        CORBA_Environment       * ev ) ;
```

Arguments

mpeg_program	*(in)* the MPEG program resource object
program_number	*(out)* the MPEG program number used in the program association table (PAT) and program map table (PMT)
map_table_pid	*(out)* the packet id (PID) which carries the program map table (PMT) for this program
conditional_access_pid	*(out)* the packet id (PID) for conditional access
stream_table	*(out)* information about each elementary stream in the program ***packet_id*** - the PID for the elementary stream ***stream_type*** - the type of elementary stream (audio, video, etc) ***association_tag*** - the association tag for the elementary stream
program_clock_ref	*(out)* which of the elementary streams in ***stream_table*** contains the program clock reference (PCR)
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about an MPEG program resource object.

Notes

1. ***program_clock_ref*** is an offset into ***stream_table***. If no PCR is present, then its value is set to -1.

2. Values of stream_type are:

ES_STREAM_TYPE_MPEG_1_VIDEO	ISO/IEC 11172 video
ES_STREAM_TYPE_MPEG_2_VIDEO	ISO/IEC 13818-2 video
ES_STREAM_TYPE_MPEG_1_AUDIO	ISO/IEC 11172 audio
ES_STREAM_TYPE_MPEG_2_AUDIO	ISO/IEC 13818-3 audio
ES_STREAM_TYPE_MHEG	ISO/IEC 13522 MHEG
ES_STREAM_TYPE_AUXILIARY	ISO/IEC 11172-1 or ISO/IEC 13818-1 auxiliary
ES_STREAM_TYPE_USER	user private stream type
ES_STREAM_TYPE_PRIVATE_SECTIONS	ISO/IEC 13818-1 private sections
ES_STREAM_TYPE_PES_PRIVATE_DATA	ISO/IEC 13818-1 elementary streams containing private data

Reference

(proprietary definition)

es_rsrc_n_isdn_info - get N-ISDN connection information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_n_isdn_conn_info
                    ( CORBA_Object          n_isdn_connection,
                    unsigned short          * b_channel,
                    CORBA_Environment       * ev ) ;
```

Arguments

n_isdn_connection	*(in)* the narrowband ISDN connection resource object
b_channel	*(out)* the number of the b-channel used for this connection
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about a narrowband ISDN connection resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_phys_chan_info - get physical channel information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_phys_chan_info
                    ( CORBA_Object          physical_channel,
                    unsigned long           * channel_id,
                    unsigned short          * direction,
                    CORBA_Environment       * ev ) ;
```

Arguments

physical_channel	*(in)* the physical channel resource object
channel_id	*(out)* channel identifier
direction	*(out)* direction of data flow
	ES_DIRECTION_DOWNSTREAM - server to client
	ES_DIRECTION_UPSTREAM - client to server
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about a physical channel resource object. Refers to a logical channel in a frequency division multiplexed medium.

Notes

1. The meaning of ***channel_id*** is private to the implementation. Examples of interpretation include frequency (in Hertz) and tuneable channel number.

Reference

(proprietary definition)

es_rsrc_q922_info - get Q.922 connection information

Synopsis - C

```
#include <spain.h>

typedef struct
        {
        unsigned short                  connection_id ;
        unsigned long                   association_tag ;
        }
        es_data_link_info_t ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        es_data_link_info_t             * _buffer ;
        }
        es_data_link_table_t ;

void    es_rsrc_q922_info
                        ( CORBA_Object          q922_connection,
                        es_data_link_table_t    ** table,
                        CORBA_Environment       * ev ) ;
```

Arguments

q922_connection	*(in)* the Q.922 connection resource object
table	*(out)* description of data link connections
	connection_id - data link connection identifier
	association_tag - the association tag for this connection
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about the connections multiplexed over a Q.922 link layer connection resource object.

Notes

(none)

Reference

(proprietary definition)

es_rsrc_tdma_up_info - get TDMA upstream connection information

Synopsis - C

```
#include <spain.h>

void    es_rsrc_tdma_up_info
                    ( CORBA_Object          client_tdma,
                    unsigned long           * start_slot_number,
                    unsigned long           * number_of_slots,
                    unsigned long           * slot_spacing,
                    unsigned long           * transport_id,
                    CORBA_Environment       * ev ) ;
```

Arguments

client_tdma	*(in)* an upstream TDMA resource object
start_slot_number	*(out)* the first TDMA slot for transmission
number_of_slots	*(out)* the number of consecutive slots available for transmission
slot_spacing	*(out)* the number of slots the session must wait after transmission before resumption
transport_id	*(out)* the transport stream on which the session should transmit data
ev	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard)

Description

Returns information about a TDMA slot assignment for upstream transmission from the resource object.

Notes

(none)

Reference

(proprietary definition)