EUROPEAN TELECOMMUNICATION STANDARD

FINAL DRAFT
pr ETS 300 777-2

September 1998

Source: MTA

Reference: DE/MTA-011057-2

ICS: 33.020

Key words: API, MHEG, multimedia, terminal

Terminal Equipment (TE); End-to-end protocols for multimedia information retrieval services; Part 2: Use of Digital Storage Media Command and Control (DSM-CC) for basic multimedia applications

ETSI

European Telecommunications Standards Institute

ETSI Secretariat

Postal address: F-06921 Sophia Antipolis CEDEX - FRANCE
Office address: 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE
Internet: secretariat@etsi.fr - http://www.etsi.fr - http://www.etsi.org

Tel.: +33 4 92 94 42 00 - Fax: +33 4 93 65 47 16

Contents

Blank page

Foreword

This final draft European Telecommunication Standard (ETS) has been produced by the Terminal Equipment (TE) Technical Committee and later the Multimedia Terminals and Applications (MTA) Project of the European Telecommunications Standards Institute (ETSI), and is now submitted for the Voting phase of the ETSI standards approval procedure.

This ETS consists of 4 parts as follows:

Part 1: "Coding of multimedia and hypermedia information for basic multimedia applications (MHEG-5)";

Part 2: "Use of Digital Storage Media Command and Control (DSM-CC) for basic multimedia applications";

Part 3: "Application Programmable Interface (API) for MHEG-5";

Part 4: "Videotex Enhanced Man Machine Interface (VEMMI) enhancements to support broadband multimedia information retrieval services".

Proposed transposition dates	
Date of latest announcement of this ETS (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

Blank page

1 Scope

This European Telecommunication Standard (ETS) specifies a profile of Digital Storage Media Command and Control (DSM-CC) for the use in basic multimedia applications. This ETS is applicable to Digital Audio Visual Council (DAVIC), version 1.2, compliant systems.

2 Normative references

This ETS incorporates by dated and undated reference, provisions from other publications. These references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

[1] DAVIC 1.2 specifications.

[2] ISO/IEC 13818-6 (1996): "Information technology; Generic coding of moving pictures and associated audio information; Part 6: Extensions for DSM-CC".

[3] ISO/IEC 13522-5 (1997): "Information technology; Coding of multimedia and hypermedia information; Part 5: Support for base-level interactive applications".

[4] ISO/IEC 13522-6 (1996): "Information technology; Coding of multimedia and hypermedia information; Part 6: Support for enhanced interactive applications".

[5] ETS 300 777-1: "Terminal Equipment (TE); End-to-end protocols for multimedia information retrieval services; Part 1: Coding of multimedia and hypermedia information for basic multimedia applications (MHEG-5)".

[6] ETS 300 777-3: "Terminal Equipment (TE); End-to-end protocols for multimedia information retrieval services; Part 3: Application Programmable Interface (API) for MHEG-5".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this ETS, the definition of the standards referenced below apply. Should any ambiguity occur, definitions of the following standards apply, in decreasing order:

- ISO/IEC 13818-6 (1996) [2];
- DAVIC 1.2 specifications [1].

Application Programmable Interface (API): A boundary across which a software application uses facilities of programming languages to invoke software services. These facilities may include procedures or operations, shared data objects and resolution of identifiers.

local application: A piece of software which is part of the (telecommunication) application and is running on the considered equipment.

3.2 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
DAVIC	Digital Audio Visual Council
DSM-CC	Digital Storage Media Command and Control
MHEG	Multimedia and Hypermedia information coding Experts Group
SI	Service Information
STU	Set Top Unit

VM Virtual Machine

4 Overview

This clause positions the API defined by this ETS in the framework of the DAVIC 1.2 specifications [1].

4.1 The DAVIC application interchange format

To deliver multimedia information to Set Top Units (STUs) in an interoperable way, applications shall use the Multimedia and Hypermedia information coding Experts Group (MHEG-5) final form interchange format, as defined by ISO/IEC 13522-5 [3]. The ASN.1 notation and encoding, as defined by ETS 300 777-1 [5], shall be used to interchange MHEG-5 objects. This format defines the semantics and the encoding of the multimedia and hypermedia objects.

To deliver program code to STUs in an interoperable way, applications shall use the MHEG-5 `InterchangedProgram` class to encapsulate Java (note) Virtual Machine (VM) code, according to the semantics and encoding defined by ISO/IEC 13522-6 [4]. Java VM classes are called from MHEG-5 objects using the MHEG-5 `Call` and `Fork` elementary actions.

NOTE: Java is a trademark of Sun Microsystems, Inc.

The Java VM code interchange unit is a Java VM class. Java VM classes shall be encoded as defined by the Class File Format section of the Java Virtual Machine specification. A Java class encapsulates data and methods that consist of sequences of instructions. The instruction set is defined by the Java Virtual Machine instruction set section of the Java Virtual Machine specification.

4.2 Core set of Java APIs

The following set of APIs are used by Java VM code in the DAVIC 1.2 specifications [1] to express access to basic functions of the STU in an interoperable way:

- the `java.lang` package;
- the `java.util` package;
- the `org.iso.mheg5` package;
- the `org.etsi.dsmccuu` package;
- the `org.etsi.dvbsi` package.

NOTE 1: The Java VM specification provides flexible mechanisms to call upon external functions whose interface is defined as a Java package. The DAVIC 1.2 specification [1] only includes a minimum core set of packages required for Java VM code to be useful in a DAVIC environment. It is anticipated that additional Java packages will be standardized at a later stage.

NOTE 2: Especially, the java.io package, although strictly speaking not necessary to the useful performance of the VM environment, is part of the Java foundation classes. It is intended that the java.io package be added to the DAVIC core set of Java APIs together with an adequate specification of its semantics in a DAVIC environment.

The `java.lang` package, as defined by the Java API documentation, consists of the minimal set of Java VM classes needed to run Java VM code, supporting the following functionality: basic data types, object, mathematic operations, security, thread management, string manipulation, exception handling.

The `java.util` package, as defined by the Java API documentation, consists of Java VM classes supporting a number of utility features common to all Java VM programs.

The `org.iso.mheg5` package, as defined by ETS 300 777-3 [6], provides Java VM code with access to and manipulation of the MHEG-5 multimedia presentation and interaction objects, i.e. access to the dynamic attributes of MHEG-5 objects and invocation of elementary actions on MHEG-5 objects.

The `org.etsi.dsmccuu` package enables Java VM code to use the DSM-CC User-to-User (U-U) interface objects for network data access. It gives access to the Core Consumer Client Application Portability Interface as defined by ISO/IEC 13818-6 [2].

The `org.etsi.dvbsi` package enables Java VM code to access information transmitted in the DAVIC Service Information (SI) stream.

5 Package org.etsi.dsmccuu

5.1 Class NameComponent

```
/**
This class represents a name component composed of an id and a kind
*/
package org.davic.net.dsmcc.uu;

public class NameComponent{

public java.lang.String id;

public java.lang.String kind;
}
```

5.2 Class Binding

```
/**
This class represents a binding composed of a name and a type
*/
package org.davic.net.dsmcc.uu;

public class Binding{

/**
Possible values for the binding type
*/
public static final byte OBJECT = 0;
public static final byte NAMING_CONTEXT = 1;

public NameComponent[] name;

public int type;
}
```

5.3 Exception NOT_FOUND

```
/**
This exception is thrown when a logical path name does not exist
*/
package org.davic.net.dsmcc.uu;

public class NOT_FOUND
   extends
      java.lang.Exception{

/**
Possible values for the exception reason
*/
public static final byte MISSING_NODE = 0;
public static final byte NOT_A_NAMING_CONTEXT = 1;
public static final byte NOT_AN_OBJECT = 2;

public byte why;

public NameComponent[] restOfName;
}
```

5.4 Exception CANNOT_PROCEED

```
/**
This exception is thrown when a Directory does not have permission to resolve
a node in a path name
The caller may attempt to perform the resolve directly at the returned
Directory
*/
package org.davic.net.dsmcc.uu;

public class CANNOT_PROCEED
   extends
      java.lang.Exception{

public Directory directory;

public NameComponent[] restOfName;
}
```

5.5 Exception INV_NAME

```
/**
This exception is thrown when a path name is incorrectly formatted
*/
package org.davic.net.dsmcc.uu;

public class INV_NAME
   extends
      java.lang.Exception{
}
```

5.6 Exception SERVICE_XFR

```
/**
This exception is thrown when a resolve operation was unsuccessful and
provides an alternate Service Domain when the requested Service can be
resolved
*/
package org.davic.net.dsmcc.uu;

public class SERVICE_XFR
   extends
      java.lang.Exception{

public byte[] serviceDomain;

public NameComponent[] pathName;

public byte[] initialContext;
}
```

5.7 Exception dsmccuuException

```
/**
This exception is the superclass for a set of exceptions which return a minor
reason and a completion status
*/
package org.davic.net.dsmcc.uu;

public class dsmccuuException
   extends
      java.lang.Exception{

/**
Possible values for the completion status
*/
public static final byte YES = 0;
public static final byte NO = 1;
public static final byte MAYBE = 2;

public short minor;

public short completed;
}
```

5.8 Exception INV_OFFSET

```
/**
This exception is thrown when the offset is outside the range of a File
content
*/
package org.davic.net.dsmcc.uu;

public class INV_OFFSET
   extends
      dsmccuuException{
}
```

5.9 Exception INV_SIZE

```
/**
This exception is thrown when the size exceeds server or network limits
*/
package org.davic.net.dsmcc.uu;

public class INV_SIZE
   extends
      dsmccuuException{
}
```

5.10 Exception READ_LOCKED

```
/**
This exception is thrown when reads are temporarily prevented for an object
*/
package org.davic.net.dsmcc.uu;

public class READ_LOCKED
   extends
      dsmccuuException{
}
```

5.11 Exception WRITE_LOCKED

```
/**
This exception is thrown when writes are temporarily prevented for an object
*/
package org.davic.net.dsmcc.uu;

public class WRITE_LOCKED
   extends
      dsmccuuException{
}
```

5.12 Exception OPEN_LIMIT

```
/**
This exception is thrown when the number of active object references is the
maximum allowed
*/
package org.davic.net.dsmcc.uu;

public class OPEN_LIMIT
   extends
      dsmccuuException{
}
```

5.13 Exception NO_AUTH

```
/**
This exception is thrown when the End-User has not provided the correct
authentication in the request
*/
package org.davic.net.dsmcc.uu;

public class NO_AUTH
   extends
      dsmccuuException{

public byte[] authData;
}
```

5.14 Exception UNK_USER

```
/**
This exception is thrown when the Principal End-User is unknown to the
Service Domain
*/
package org.davic.net.dsmcc.uu;

public class UNK_USER
   extends
      dsmccuuException{
}
```

5.15 Exception BAD_COMPAT_INFO

```
/**
This exception is thrown in response to an operation which carries a
DownloadInfoRequest with an unrecognized CompatibilityDescriptor
*/
package org.davic.net.dsmcc.uu;

public class BAD_COMPAT_INFO
   extends
      dsmccuuException{
}
```

5.16 Exception NO_RESUME

```
/**
This exception is thrown when the previous application saved state cannot be
recovered
*/
package org.davic.net.dsmcc.uu;

public class NO_RESUME
   extends
      dsmccuuException{
}
```

5.17 Exception NO_SUSPEND

```
/**
This exception is thrown when the object cannot save the application state
*/
package org.davic.net.dsmcc.uu;

public class NO_SUSPEND
   extends
      dsmccuuException{
}
```

5.18 Exception MPEG_DELIVERY

```
/**
This exception is thrown when the Server is unable to deliver a mumtimedia
object over an MPEG stream
*/
package org.davic.net.dsmcc.uu;

public class MPEG_DELIVERY
   extends
      dsmccuuException{
}
```

5.19 Exception BAD_SCALE

```
/**
This exception is thrown when the Scale value is incorrect
*/
package org.davic.net.dsmcc.uu;

public class BAD_SCALE
   extends
      dsmccuuException{
}
```

5.20 Exception BAD_START

```
/**
This exception is thrown when the Stream start time does not exist
*/
package org.davic.net.dsmcc.uu;

public class BAD_START
   extends
      dsmccuuException{
}
```

5.21 Exception BAD_STOP

```
/**
This exception is thrown when the Stream stop time does not exist
*/
package org.davic.net.dsmcc.uu;

public class BAD_ STOP
   extends
      dsmccuuException{
}
```

5.22 Interface Base

```
/**
This interface shall be implemented to provide common operations for deletion
of DSM-CC object references
*/
package org.davic.net.dsmcc.uu;

interface Base{

/**
To indicate that access to the object is no longer required
*/
public void close(
);
}
```

5.23 Interface Access

```
/**
This interface shall be implemented to provide access to the lock attributes
*/
package org.davic.net.dsmcc.uu;

interface Access{

/**
To get the lock attributes (the read lock is returned in the first element of
the array, the write lock in the second element)
*/
public boolean[] getLock(
);

/**
To set the lock attributes (the read lock is specified in the first element
of the array, the write lock in the second element)
*/
public void setLock(
   boolean[] lock
);
}
```

5.24 Class Stream

```
/**
This class represents the DSM Stream interface
*/
package org.davic.net.dsmcc.uu;

public class Stream
   implements
      Base, Access{

/**
Possible values for the stream mode
*/
public static final byte OPEN = 0;
public static final byte PAUSE = 1;
public static final byte TRANSPORT = 2;
public static final byte TRANSPORT_PAUSE = 3;
public static final byte SEARCH_TRANSPORT = 4;
public static final byte SEARCH_TRANSPORT_PAUSE = 5;
public static final byte PAUSE_SEARCH_TRANSPORT = 6;
public static final byte END_OF_STREAM = 7;
public static final byte PRE_SEARCH_TRANSPORT = 8;
public static final byte PRE_SEARCH_TRANSPORT_PAUSE = 9;

/**
Base.close implementation
*/
public void close(
)
{
// actual code shall be inserted here
}

/**
Access.getLock implementation
*/
public boolean[] getLock(
)
{
// actual code shall replace the following statement
return(null);
}

/**
Access.setLock implementation
*/
public void setLock(
   boolean[] lock
)
{
// actual code shall be inserted here
}

/**
To cause the video server to stop sending the stream
rStop gives the position at which the pause shall occur; this position is
expressed in seconds,microseconds and is stored in an array (first element
for the seconds, second element for the microseconds)
*/
public void pause(
   int[] rStop
) throws
   MPEG_DELIVERY, BAD_STOP
{
// actual code shall be inserted here
}
```

```
/**
To cause the video server to resume sending the stream
rStart gives the position at which to resume
rScale gives the scale at which to resume; this scale is expressed with a
numerator (possibly negative) and a denominator and is stored in an array
(first element for the numerator, second element for the denominator)
*/
public void resume(
   int[] rStart,
   int[] rScale
) throws
   MPEG_DELIVERY, BAD_START, BAD_SCALE
{
// actual code shall be inserted here
}

/**
To get the current position of a stream in progress
rPosition contains the application's estimation of the current position
*/
public int[] getPosition(
   int[] rPosition
) throws
   MPEG_DELIVERY
{
// actual code shall replace the following statement
return(null);
}

/**
To get the current scale of a stream in progress
*/
public int[] getScale(
) throws
   MPEG_DELIVERY
{
// actual code shall replace the following statement
return(null);
}

/**
To get the current mode of a stream in progress
*/
public byte getMode(
) throws
   MPEG_DELIVERY
{
// actual code shall replace the following statement
return((byte) 0);
}

/**
To reset the Stream state machine
*/
public void reset(
)
{
// actual code shall be inserted here
}

/**
```

```
To cause the video server to jump and resume from a start position when a
stop position is reached
*/
public void jump(
   int[] rStart,
   int[] rStop,
   int[] rScale
) throws
   MPEG_DELIVERY, BAD_START, BAD_STOP, BAD_SCALE
{
// actual code shall be inserted here
}

/**
To cause the video server to play from a start position until a stop position
*/
public void play(
   int[] rStart,
   int[] rStop,
   int[] rScale
) throws
   MPEG_DELIVERY, BAD_START, BAD_STOP, BAD_SCALE
{
// actual code shall be inserted here
}
}
```

5.25 Class File

```
/**
This class represents the DSM Stream interface
*/
package org.davic.net.dsmcc.uu;

public class File
   implements
      Base, Access{

/**
Base.close implementation
*/
public void close(
)
{
// actual code shall be inserted here
}

/**
Access.getLock implementation
*/
public boolean[] getLock(
)
{
// actual code shall replace the following statement
return(null);
}

/**
Access.setLock implementation
*/
public void setLock(
   boolean[] lock
)
{
// actual code shall be inserted here
}
```

```
/**
To read data
*/
public byte[] read(
   long aOffset,
   int aSize,
   boolean aReliable
) throws
   INV_OFFSET, INV_SIZE, READ_LOCKED
{
// actual code shall replace the following statement
return(null);
}

/**
To get the content
*/
public byte[] getContent(
)
{
// actual code shall replace the following statement
return(null);
}

/**
To set the content
*/
public void setContent(
   byte[] content
)
{
// actual code shall be inserted here
}

/**
To write data
*/
public void write(
   long aOffset,
   int aSize,
   byte[] rData
) throws
   INV_OFFSET, INV_SIZE, WRITE_LOCKED
{
// actual code shall be inserted here
}
}
```

5.26 Class BindingIterator

```
/**
This class allows to browse through a set of bindings
*/
package org.davic.net.dsmcc.uu;

public class BindingIterator
   implements
      java.util.Enumeration{

/**
To know if there is more elements
(this method implements java.util.Enumeration.hasMoreElement())
*/
public boolean hasMoreElements(
)
{
// actual code shall replace the following statement
return (false);
}

/**
To get the next element
(this method implements java.util.Enumeration.nextElement())
*/
public Object nextElement(
)
{
// actual code shall replace the following statement
return (null);
}

/**
To discard any server-side storage associated with the iterator and make it
no longer valid to access
*/
public void destroy(
)
{
// actual code shall be inserted here
}
}
```

5.27 Class Directory

```
/**
This class represents the DSM Directory interface
*/
package org.davic.net.dsmcc.uu;

public class Directory
   implements
      Access{

/**
Possible values for the path type
*/
public static final byte DEPTH = 0;
public static final byte BREADTH = 1;

/**
Access.getLock implementation
*/
public boolean[] getLock(
)
{
// actual code shall replace the following statement
return(null);
}

/**
Access.setLock implementation
*/
public void setLock(
   boolean[] lock
)
{
// actual code shall be inserted here
}

/**
To get the list of bindings
*/
public BindingIterator list(
)
{
// actual code shall replace the following statement
return(null);
}

/**
To resolve a name
*/
public Object resolve(
   NameComponent[] name
) throws
   NOT_FOUND, CANNOT_PROCEED, INV_NAME
{
// actual code shall replace the following statement
return(null);
}

/**
```

```
To resolve several names
*/
public Object[] open(
   byte aPathType,
   NameComponent[] name,
   boolean[] process
) throws
   OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR,
   NOT_FOUND, CANNOT_PROCEED, INV_NAME
{
// actual code shall replace the following statement
return(null);
}

/**
To indicate that access to the Directory is no longer required
*/
public void close(
)
{
// actual code shall be inserted here
}

/**
To get one (or more) attibute values of a Directory entry
*/
public Object[] get(
   byte aPathType,
   NameComponent[] name,
   boolean[] process
) throws
   NO_AUTH, UNK_USER, SERVICE_XFR,
   NOT_FOUND, CANNOT_PROCEED, INV_NAME
{
// actual code shall replace the following statement
return(null);
}
}
```

5.28 Class Session

```
/**
This class represents the DSM Session interface
*/
package org.davic.net.dsmcc.uu;

public class Session{

/**
Constructor
*/
public Session(
)
{
// actual code shall be inserted here
}

/**
To establish a Session context with a ServiceGateway
*/
public Object[] attach(
   byte[] serviceDomain,
   NameComponent[] pathName,
   byte[] userContext
) throws
   OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR, BAD_COMPAT_INFO, NO_RESUME,
   NOT_FOUND, CANNOT_PROCEED, INV_NAME
```

```
{
// actual code shall replace the following statement
return(null);
}
}
```

5.29 Class ServiceGateway

```
/**
This class represents the DSM ServiceGateway interface
*/
package org.davic.net.dsmcc.uu;

public class ServiceGateway
   extends
      Directory{

/**
To disconnect from a ServiceGateway and all objects of a Session
*/
public byte[] detach(
   boolean aSuspend
) throws
   NO_SUSPEND
{
// actual code shall replace the following statement
return(null);
}
}
```

5.30 Class First

```
/**
This class represents the DSM First interface
*/
package org.davic.net.dsmcc.uu;

public class First{

/*
Constructor
*/
public First(
)
{
// actual code shall be inserted here
}

/*
To obtain the object reference of the ServiceGateway for the current Session
*/
public ServiceGateway root(
)
{
// actual code shall replace the following statement
return(null);
}

/*
```

```
To obtain the object reference of the current Session's Primary Service
*/
public Object service(
)
{
// actual code shall replace the following statement
return(null);
}
}
```

6 Example

This clause gives - for information only - an usage example for the API specified in this ETS.

This example illustrates how to attach to a ServiceGateway, to use the available Services, to free a Service and to detach from the ServiceGateway.

```
/*
 *
 * Example
 *
 */

class Example{

void main()
{
/*
Declaration of an array to store object references; the first element will be
used to store the ServiceGateway object reference, the second one will be
used to store the first Service object reference
*/
Object objRefs[] = new Object[2];

/*
Has a Session already been established?
*/
if(true)
{
   /* NO */
   /*
   Create a local Session object
   */
   Session localSessionObjRef = new Session();

   /*
   Attach to a ServiceGateway
   */
   NameComponent[] pathName = new NameComponent[1];
   pathName[0].id = "myGateway";
   pathName[0].kind = "srg";
   try
   {
      objRefs = localSessionObjRef.attach(null,pathName,null);
   }
   catch(OPEN_LIMIT e)
   {
      /* ... */
   }
   catch(NO_AUTH e)
   {
      /* ... */
   }
```

```
   catch(UNK_USER e)
   {
      /* ... */
   }
   catch(SERVICE_XFR e)
   {
      /* ... */
   }
   catch(BAD_COMPAT_INFO e)
   {
      /* ... */
   }
   catch(NO_RESUME e)
   {
      /* ... */
   }
   catch(NOT_FOUND e)
   {
      /* ... */
   }
   catch(CANNOT_PROCEED e)
   {
      /* ... */
   }
   catch(INV_NAME e)
   {
      /* ... */
   }
}
else
{
   /* YES */
   /*
   Create a local First object
   */
   First localFirstObjRef = new First();

   /*
   Get the object reference of the ServiceGateway
   */
   objRefs[1] = localFirstObjRef.root();

   /*
   Get the object reference of the first Service
   */
   objRefs[2] = localFirstObjRef.service();
}

/*
If the first Service is a Directory, browse through
*/
if(true)
{
   /*
   Get the list of bindings and iterate through the list
   */
   BindingIterator bi = ((Directory) objRefs[2]).list();
   while(bi.hasMoreElements())
   {
      Binding b = (Binding) bi.nextElement();
      /*
      Process the name (in b.name) and the type (in b.type) of the binding
      */
      /* ... */
   }

   /*
   or
```

```
Resolve a name (the name of a Stream called "myStream.mpeg" in a Directory
called "streams")
*/
NameComponent[] name = new NameComponent[2];
name[0].id = "streams";
name[0].kind = "dir";
name[1].id = "myStream.mpeg";
name[1].kind = "str";
try
{
   Stream stream = (Stream) ((Directory) objRefs[2]).resolve(name);
}
catch(NOT_FOUND e)
{
   /* ... */
}
catch(CANNOT_PROCEED e)
{
   /* ... */
}
catch(INV_NAME e)
{
   /* ... */
}

/*
or
Resolve several names (the name of a Directory called "streams" and the
name of a Stream called "myStream.mpeg" within that directory)
*/
boolean[] process = new boolean[2];
name[0].id = "streams";
name[0].kind = "dir";
process[0] = true;
name[1].id = "myStream.mpeg";
name[1].kind = "str";
process[1] = true;
try
{
   Object[2] someObjRefs =
   ((Directory) objRefs[2]).open(Directory.DEPTH,name,process);
}
catch(OPEN_LIMIT e)
{
   /* ... */
}
catch(NO_AUTH e)
{
   /* ... */
}
catch(UNK_USER e)
{
   /* ... */
}
catch(SERVICE_XFR e)
{
   /* ... */
}
catch(BAD_COMPAT_INFO e)
{
   /* ... */
}
catch(NO_RESUME e)
{
   /* ... */
}
```

```
   catch(NOT_FOUND e)
   {
      /* ... */
   }
   catch(CANNOT_PROCEED e)
   {
      /* ... */
   }
   catch(INV_NAME e)
   {
      /* ... */
   }
}

/*
If the first Service is a File, read the first 128 bytes
*/
if(true)
{
   try
   {
      byte[] rData = new byte[128];
      rData = ((File) objRefs[2]).read(0,128,true);
   }
   catch(INV_OFFSET e)
   {
      /* ... */
   }
   catch(INV_SIZE e)
   {
      /* ... */
   }
   catch(READ_LOCKED e)
   {
      /* ... */
   }
}

/*
If the first Service is a Stream, start to play it from the beginning at
normal rate
*/
if(true)
{
   int rStart[] = new int[2];
   rStart[0] = 0x80000000;
   rStart[1]= 0;
   int rScale[] = new int[2];
   rScale[0] = 1;
   rScale[1] = 1;
   try
   {
      ((Stream) objRefs[2]).resume(rStart, rScale);
   }
   catch(MPEG_DELIVERY e)
   {
      /* ... */
   }
   catch(BAD_START e)
   {
      /* ... */
   }
   catch(BAD_SCALE e)
   {
   /* ... */
   }
}
```

```
/*
Free the Service
*/
((Directory) objRefs[2]).close();
/*
or
*/
((File) objRefs[2]).close();
/*
or
*/
((Stream) objRefs[2]).close();

/*
Detach from the ServiceGateway, save the context
*/
try
{
   byte[] savedContext = ((ServiceGateway) objRefs[1]).detach(true);
}
catch(NO_SUSPEND e)
{
   /* ... */
}
}
}
```

History

Document history			
August 1997	Public Enquiry	PE 9750:	1997-08-15 to 1997-12-12
September 1998	Vote	V 9848:	1998-09-29 to 1998-11-27