INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.176

(02/98)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Application Programming Interface (API) for Digital Storage Media Command and Control (DSM-CC)

ITU-T Recommendation T.176

(Previously CCITT Recommendation)

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION T.176

APPLICATION PROGRAMMING INTERFACE (API) FOR DIGITAL STORAGE MEDIA COMMAND AND CONTROL (DSM-CC)

Summary

This Recommendation specifies the Application Programming Interface (API) of DSM-CC for the use in basic multimedia applications.

Source

ITU-T Recommendation T.176 was prepared by ITU-T Study Group 16 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 6th of February 1998.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation T.176

APPLICATION PROGRAMMING INTERFACE (API) FOR DIGITAL STORAGE MEDIA COMMAND AND CONTROL (DSM-CC)

(Geneva, 1998)

1 Scope

This Recommendation specifies the Application Programming Interface (API) of DSM-CC for the use in basic multimedia applications. This Recommendation is applicable to DAVIC compliant systems.

2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[1] 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*.

[2] ISO/IEC DIS 13818-6, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extension for digital storage media command and control*.

[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 DIS 13522-6, *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).*

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this Recommendation the definition of ISO/IEC DIS 13818-6 [2] apply.

This Recommendation defines the following terms:

3.1.1 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.

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

3.2 Abbreviations

This Recommendation uses the following abbreviations

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

The following clause positions the API defined by this Recommendation in the framework of the DAVIC specifications.

4.1 The DAVIC application interchange format

To deliver multimedia information to STUs in an interoperable way, applications shall use the 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 [4], 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[1] VM code, according to the semantics and encoding defined by ISO/IEC DIS 13522-6 [5]. Java VM classes are called from MHEG-5 objects using the MHEG-5 `Call` and `Fork` elementary actions.

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 1.1 [1] specifications to express access to basic functions of the STU in an interoperable way:

– the `java.lang` package;
– the `java.util` package;
– the `iso.mheg5` package;
– the `davic.dsmccuu` package;
– the `etsi.si` package.

---

[1] Java is a trademark or a registered trademark of Sun Microsystems, Inc.

NOTE 1 – The Java VM specification provides flexible mechanisms to call upon external functions whose interface is defined as a Java package. The DAVIC 1.1 specification 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 standardised 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 `iso.mheg5` package, as defined by ETS 300 777-3 [1], 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 `davic.dsmccuu` and the associated `davic.CosNaming` packages enable Java VM code to use the DSM-CC U-U interface objects for network data access.

The `davic.dsmccuu` and the associated `davic.CosNaming` packages give access to a subset of the DSM-CC U-U API defined by ISO/IEC DIS 13818-6. This subset consists of:

– the `list` and `resolve` operations of the `NamingContext` abstract interface;
– the `close` and `destroy` operations of the `Base` abstract interface;
– the `next_one` and `next_n` operations of the `BindingIterator` instanciable interface;
– the `open` and `close` operations of the `Directory` instanciable interface;
– the `read` and `write` operations as well as the `ContentSize` read-only attribute of the `File` instanciable interface;
– the `attach` and `detach` operations of the `Session` instanciable interface;
– the `SessionGateway` instanciable interface.

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

5 Package davic.CosNaming

5.1 Class davic.CosNaming.NameComponent

```
package davic.CosNaming;

public class NameComponent {
        public String id;
        public String kind;
}
```

5.2 Class davic.CosNaming.Binding

```
package davic.CosNaming;

public class Binding {
        // constant declarations for the "binding_type" attribute
        public static final short nobject = 0;
        public static final short ncontext = 1;

        public NameComponent[] binding_name;
        public int binding_type;
}
```

5.3 Exception davic.CosNaming.NotFound

```
package davic.CosNaming;

public class NotFound extends Exception{
        // constant declarations for the "why" attribute
        public static final short missing_node = 0;
        public static final short not_context = 1;
        public static final short not_object = 2;

        public int why;
        public NameComponent[] rest_of_name;
}
```

5.4 Exception davic.CosNaming.CannotProceed

```
package davic.CosNaming;

public class CannotProceed extends Exception{
        public NamingContext cxt;
        public NameComponent[] rest_of_name;
}
```

5.5 Exception davic.CosNaming.InvalidName

```
package davic.CosNaming;

public class InvalidName extends Exception{
}
```

5.6 Class davic.CosNaming.BindingIterator

```
package davic.CosNaming;

public class BindingIterator {
        public boolean next_one(
                Binding b
        )
        {
        // actual code shall be inserted here
        return true;
        }
```

```
        public void next_n(
                int how_many,
                Binding[] bl
        )
        {
        // actual code shall be inserted here
        }

        public void destroy()
        {
        // actual code shall be inserted here
        }
}
```

5.7 Interface davic.CosNaming.NamingContext

```
package davic.CosNaming;

public interface NamingContext {
        public void list(
                int how_many,
                Binding[] bl,
                BindingIterator bi
        );

        public Object resolve(
                NameComponent[] n
        )throws NotFound, CannotProceed, InvalidName;
}
```

6 Package davic.dsmccuu

6.1 Class davic.dsmccuu.Step

```
package davic.dsmccuu;

import davic.CosNaming.*;

public class Step {
        public NameComponent name;
        public boolean process;
}
```

6.2 Exception davic.dsmccuu.SERVICE_XFR

```
package davic.dsmccuu;

import davic.CosNaming.*;

public class SERVICE_XFR extends Exception {
        // service location
        public byte[] serviceDomain;
        public NameComponent[] pathName;
        public byte[] initialContext;
}
```

6.3 Exception davic.dsmccuu.dsmccuuException

```
package davic.dsmccuu;

public class dsmccuuException extends Exception {
        public short minor;
        public short completed;
}
```

6.4 Exception davic.dsmccuu.INV_OFFSET

```
package davic.dsmccuu;

public class INV_OFFSET extends dsmccuuException {
}
```

6.5 Exception davic.dsmccuu.INV_SIZE

```
package davic.dsmccuu;

public class INV_SIZE extends dsmccuuException {
}
```

6.6 Exception davic.dsmccuu.READ_LOCKED

```
package davic.dsmccuu;

public class READ_LOCKED extends dsmccuuException {
}
```

6.7 Exception davic.dsmccuu.WRITE_LOCKED

```
package davic.dsmccuu;

public class WRITE_LOCKED extends dsmccuuException {
}
```

6.8 Exception davic.dsmccuu.OPEN_LIMIT

```
package davic.dsmccuu;

public class OPEN_LIMIT extends dsmccuuException {
}
```

6.9 Exception davic.dsmccuu.NO_AUTH

```
package davic.dsmccuu;

public class NO_AUTH extends dsmccuuException {
        public byte[] authData;
}
```

6.10 Exception davic.dsmccuu.UNK_USER

```
package davic.dsmccuu;

public class UNK_USER extends dsmccuuException {
}
```

6.11 Exception davic.dsmccuu.BAD_COMPAT_INFO

```
package davic.dsmccuu;

public class BAD_COMPAT_INFO extends dsmccuuException {
}
```

6.12 Exception davic.dsmccuu.NO_RESUME

```
package davic.dsmccuu;

public class NO_RESUME extends dsmccuuException {
}
```

6.13 Exception davic.dsmccuu.NO_SUSPEND

```
package davic.dsmccuu;

public class NO_SUSPEND extends dsmccuuException {
}
```

6.14 Interface davic.dsmccuu.Base

```
package davic.dsmccuu;

interface Base {
        public void close();

        public void destroy();
}
```

6.15 Class davic.dsmccuu.File

```
package davic.dsmccuu;

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

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

        public int[] getContentSize()
        {
        // actual code shall be inserted here
        return null;
        }

        public void read(
                int[] aOffset,
                int aSize,
                boolean aReliable,
                byte[] rData
```

```
        ) throws
                INV_OFFSET, INV_SIZE, READ_LOCKED
        {
        // actual code shall be inserted here
        }

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

6.16 Class davic.dsmccuu.Directory

```
package davic.dsmccuu;

import davic.CosNaming.*;

public class Directory implements NamingContext {
        // NamingContext.list implementation
        public void list(
                int how_many,
                Binding[] bl,
                BindingIterator bi
        )
        {
        // actual code shall be inserted here
        }

        // NamingContext.resolve implementation
        public Object resolve(
                NameComponent[] n
        )throws NotFound, CannotProceed, InvalidName
        {
        // actual code shall be inserted here
        return null;
        }

        public void open(
                char aPathType,
                Step[] rPathStep,
                Object[] resolvedRefs
        ) throws
                OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR,
                NotFound, CannotProceed, InvalidName
        {
        // actual code shall be inserted here
        }

        public void close()
        {
        // actual code shall be inserted here
        }
}
```

6.17 Interface davic.dsmccuu.SessionI

```
package davic.dsmccuu;

import davic.CosNaming.*;

interface SessionI {
        public void attach(
                byte[] serviceDomain,
                NameComponent[] pathName,
                byte[] userContext,
                Object[] resolvedRefs
        ) throws
                OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR, BAD_COMPAT_INFO, NO_RESUME,
                NotFound, CannotProceed, InvalidName;

        public void detach(
                boolean aSuspend,
                byte[] savedContext
        ) throws
                NO_SUSPEND;
}
```

6.18 Class davic.dsmccuu.Session

```
package davic.dsmccuu;

public class Session implements SessionI {
        // SessionI.attach implementation
        public void attach(
                byte[] serviceDomain,
                NameComponent[] pathName,
                byte[] userContext,
                Object[] resolvedRefs
        ) throws
                OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR, BAD_COMPAT_INFO, NO_RESUME,
                NotFound, CannotProceed, InvalidName
        {
        // actual code shall be inserted here
        }

        // SessionI.detach implementation
        public void detach(
                boolean aSuspend,
                byte[] savedContext
        ) throws
                NO_SUSPEND
        {
        // actual code shall be inserted here
        }
}
```

6.19 Class davic.dsmccuu.SessionGateway

```
package davic.dsmccuu;

public class SessionGateway extends Directory implements SessionI {
        // SessionI.attach implementation
        public void attach(
                byte[] serviceDomain,
                NameComponent[] pathName,
                byte[] userContext,
                Object[] resolvedRefs
        ) throws
                OPEN_LIMIT, NO_AUTH, UNK_USER, SERVICE_XFR, BAD_COMPAT_INFO, NO_RESUME,
                NotFound, CannotProceed, InvalidName
        {
        // actual code shall be inserted here
        }

        // SessionI.detach implementation
        public void detach(
                boolean aSuspend,
                byte[] savedContext
        ) throws
                NO_SUSPEND
        {
        // actual code shall be inserted here
        }
}
```

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages