

---

---

**Information technology — Open Systems  
Interconnection — Systems Management:  
Command sequencer for Systems  
Management**

*Technologies de l'information — Interconnexion de systèmes ouverts  
(OSI) — Gestion-systèmes: Séquenceur de commande pour la  
gestion-systèmes*



**Contents***Page*

|      |                                                                                       |    |
|------|---------------------------------------------------------------------------------------|----|
| 1    | Scope .....                                                                           | 1  |
| 2    | Normative references .....                                                            | 1  |
| 2.1  | Identical Recommendations   International Standards .....                             | 1  |
| 2.2  | Paired Recommendations   International Standards equivalent in technical content..... | 2  |
| 3    | Definitions.....                                                                      | 3  |
| 3.1  | Basic Reference Model definitions.....                                                | 3  |
| 3.2  | Service convention definitions .....                                                  | 3  |
| 3.3  | Management framework definitions.....                                                 | 3  |
| 3.4  | Systems management overview definitions.....                                          | 3  |
| 3.5  | Common management information service definitions.....                                | 3  |
| 3.6  | Additional definitions .....                                                          | 3  |
| 4    | Abbreviations .....                                                                   | 4  |
| 5    | Conventions .....                                                                     | 4  |
| 6    | Requirements .....                                                                    | 4  |
| 7    | Model .....                                                                           | 5  |
| 7.1  | Model description .....                                                               | 5  |
| 7.2  | Triggering process and reporting results .....                                        | 7  |
| 7.3  | Management of command sequencer .....                                                 | 8  |
| 7.4  | Scheduling of the command sequencer .....                                             | 10 |
| 7.5  | Access control.....                                                                   | 10 |
| 8    | Generic definitions.....                                                              | 10 |
| 8.1  | Managed objects .....                                                                 | 10 |
| 8.2  | Generic notifications.....                                                            | 16 |
| 8.3  | Generic actions .....                                                                 | 17 |
| 9    | Services.....                                                                         | 17 |
| 9.1  | Introduction .....                                                                    | 17 |
| 9.2  | Initiation, Termination, Modification and Retrieval Services .....                    | 17 |
| 9.3  | Notification services .....                                                           | 17 |
| 9.4  | Action services.....                                                                  | 19 |
| 10   | Functional units.....                                                                 | 21 |
| 11   | Protocols and abstract syntax .....                                                   | 21 |
| 11.1 | Abstract syntax .....                                                                 | 21 |
| 11.2 | Attributes .....                                                                      | 21 |
| 11.4 | Notifications .....                                                                   | 22 |
| 11.5 | Actions.....                                                                          | 23 |
| 11.6 | Negotiation of functional units .....                                                 | 23 |
| 12   | Relationship with other functions .....                                               | 23 |

© ISO/IEC 1998

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

|         |                                                                              |     |
|---------|------------------------------------------------------------------------------|-----|
| 13      | Conformance .....                                                            | 23  |
| 13.1    | General conformance class requirements .....                                 | 24  |
| 13.2    | Dependent conformance class requirements .....                               | 24  |
| 13.3    | Conformance to support managed object definitions .....                      | 24  |
| Annex A | – Definition of Management Information .....                                 | 25  |
| A.1     | Managed object class definitions .....                                       | 25  |
| A.2     | Packaging definitions .....                                                  | 27  |
| A.3     | Behaviour definitions .....                                                  | 29  |
| A.4     | Attribute definitions .....                                                  | 30  |
| A.5     | Notification definitions .....                                               | 32  |
| A.6     | Action definitions .....                                                     | 33  |
| A.7     | Name binding definitions .....                                               | 33  |
| A.8     | ASN.1 definitions .....                                                      | 35  |
| Annex B | – General Relationship Model .....                                           | 38  |
| Annex C | – Management Information Definitions for Event Discrimination Counting ..... | 44  |
| C.1     | Managed object class .....                                                   | 44  |
| C.2     | Package .....                                                                | 44  |
| C.3     | Attribute .....                                                              | 45  |
| Annex D | – cmisScript Management Support Object Class .....                           | 46  |
| D.1     | Attributes .....                                                             | 46  |
| D.2     | Definitions .....                                                            | 46  |
| D.3     | getCmisScript .....                                                          | 46  |
| D.4     | setCmisScript .....                                                          | 47  |
| D.5     | actionCmisScript .....                                                       | 47  |
| D.6     | createCmisScript .....                                                       | 47  |
| D.7     | deleteCmisScript .....                                                       | 48  |
| D.8     | Services .....                                                               | 48  |
| D.9     | GDMO template .....                                                          | 48  |
| Annex E | – CMIP_CS managed object class .....                                         | 54  |
| E.1     | cmipCS .....                                                                 | 54  |
| Annex F | – Systems Management Scripting Language (SMSL) .....                         | 55  |
| F.1     | Mapping GDMO onto SMSL .....                                                 | 55  |
| F.2     | SMSL Built-in functions .....                                                | 55  |
| F.3     | Set functions for SMSL lists .....                                           | 55  |
| F.4     | SMSL mathematical functions .....                                            | 56  |
| F.5     | SMSL process synchronization .....                                           | 56  |
| F.6     | SMSL shared global channels .....                                            | 56  |
| F.7     | SMSL data types and objects .....                                            | 56  |
| F.8     | SMSL variables .....                                                         | 57  |
| F.9     | SMSL predefined constants .....                                              | 58  |
| F.10    | SMSL string literals .....                                                   | 58  |
| F.11    | SMSL lists .....                                                             | 59  |
| F.12    | SMSL simple statements .....                                                 | 59  |
| F.13    | SMSL operators .....                                                         | 59  |
| F.14    | The SMSL core scripting language .....                                       | 62  |
| Annex G | – SMSL support functions .....                                               | 81  |
| Annex H | – MOCS proforma .....                                                        | 122 |
| H.1     | Statement of conformance to the basicSpawnerClass object class .....         | 122 |
| H.2     | Statement of conformance to the commandSequencer object class .....          | 124 |
| H.3     | Statement of conformance to the generalstringScript object class .....       | 127 |

H.4 Statement of conformance to the launchPad object class ..... 129

H.5 Statement of conformance to the asynchronousLaunchPad object class ..... 133

H.6 Statement of conformance to the synchronousLaunchPad object class..... 136

H.7 Statement of conformance to the launchScript object class..... 140

H.8 Statement of conformance to the scriptReferencer object class..... 142

H.9 Statement of conformance to the thread object class..... 143

H.10 Statement of conformance to the suspendableThread object class ..... 146

H.11 Statement of conformance to the eventDiscriminationCounter object class..... 151

H.12 Statement of conformance to the cmipCS object class..... 157

H.13 Statement of conformance to the cmisScript object class..... 161

H.14 Statement of conformance to the getCmisScript object class ..... 162

H.15 Statement of conformance to the setCmisScript object class..... 164

H.16 Statement of conformance to the actionCmisScript object class ..... 166

H.17 Statement of conformance to the createCmisScript object class ..... 168

H.18 Statement of conformance to the deleteCmisScript object class ..... 170

## Foreword

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

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

International Standard ISO/IEC 10164-21 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 33, *Distributed application services*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.753.

ISO/IEC 10164 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — Systems Management*:

- *Part 1: Object management function*
- *Part 2: State management function*
- *Part 3: Attributes for representing relationships*
- *Part 4: Alarm reporting function*
- *Part 5: Event report management function*
- *Part 6: Log control function*
- *Part 7: Security alarm reporting function*
- *Part 8: Security audit trail function*
- *Part 9: Objects and attributes for access control*
- *Part 10: Usage metering function for accounting purposes*
- *Part 11: Metric objects and attributes*
- *Part 12: Test management function*
- *Part 13: Summarization function*
- *Part 14: Confidence and diagnostic test categories*
- *Part 15: Scheduling function*
- *Part 16: Management knowledge management function*
- *Part 17: Change over function*
- *Part 18: Software management function*
- *Part 19: Management domain and management policy management functions*
- *Part 20: Time management function*
- *Part 21: Command sequencer for Systems Management*
- *Part 22: Response time monitoring function*

Annexes A, B, D and F to H form an integral part of this part of ISO/IEC 10164. Annexes C and E are for information only.

## INTERNATIONAL STANDARD

## ITU-T RECOMMENDATION

# INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – SYSTEMS MANAGEMENT: COMMAND SEQUENCER FOR SYSTEMS MANAGEMENT

## 1 Scope

This Recommendation | International Standard defines a Systems Management Function which may be used by an application process in a centralized or decentralized management environment to interact for the purpose of systems management, as defined by CCITT Rec. X.700 | ISO/IEC 7498-4. This Recommendation | International Standard defines the Command Sequencer which consists of generic definitions, services and functional units. This function is positioned in the application layer of ITU-T Rec. X.200 | ISO/IEC 7498-1 and is defined according to the model provided by ISO 9545. The role of systems management functions is described by ITU Rec. X.701 | ISO/IEC 10040.

This Recommendation | International Standard:

- establishes user requirements for the Command Sequencer;
- establishes models that relate the services provided by the function to user requirements;
- defines the services provided by the function;
- specifies the protocol that is necessary in order to provide the services;
- defines the relationship between the services and SMI operations and notifications;
- defines relationships with other systems management functions;
- specifies conformance requirements;
- defines a scripting language for use in the command sequencer environment.

This Recommendation | International Standard does not:

- define the nature of any implementation intended to provide the Command Sequencer;
- specify the manner in which management is accomplished by the use of the Command Sequencer;
- define the nature of any instructions which result in the use of the Command Sequencer;
- specify the services necessary for the establishment, normal, abnormal release of management associations.

## 2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

### 2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*.
- ITU-T Recommendation X.210 (1993) | ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: Conventions for the definition of OSI services*.

- CCITT Recommendation X.701 (1992) | ISO/IEC 10040:1992, *Information technology – Open Systems Interconnection – Systems management overview.*
- ITU-T Recommendation X.710 (1997) | ISO/IEC 9595:1998, *Information technology – Open Systems Interconnection – Common management information service.*
- ITU-T Recommendation X.711 (1997) | ISO/IEC 9596-1:1998, *Information technology – Open Systems Interconnection – Common management information protocol: Specification.*
- CCITT Recommendation X.721 (1992) | ISO/IEC 10165-2:1992, *Information technology – Open Systems Interconnection – Structure of management information: Definition of management information.*
- CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of managed objects.*
- ITU-T Recommendation X.724 (1996) | ISO/IEC 10165-6:1997, *Information technology – Open Systems Interconnection – Structure of management information: Requirements and guidelines for implementation conformance statement proformas associated with OSI management.*
- ITU-T Recommendation X.725 (1995) | ISO/IEC 10165-7:1996, *Information technology – Open Systems Interconnection – Structure of management information: General relationship model.*
- CCITT Recommendation X.730 (1992) | ISO/IEC 10164-1:1993, *Information technology – Open Systems Interconnection – Systems management: Object management function.*
- CCITT Recommendation X.731 (1992) | ISO/IEC 10164-2:1992, *Information technology – Open Systems Interconnection – Systems management: State management function.*
- CCITT Recommendation X.733 (1992) | ISO/IEC 10164-4:1992, *Information technology – Open Systems Interconnection – Systems management: Alarm reporting function.*
- CCITT Recommendation X.734 (1992) | ISO/IEC 10164-5:1993, *Information technology – Open Systems Interconnection – Systems management: Event report management function.*
- CCITT Recommendation X.735 (1992) | ISO/IEC 10164-6:1993, *Information technology – Open Systems Interconnection – Systems management: Log control function.*
- ITU-T Recommendation X.739 (1993) | ISO/IEC 10164-11:1994, *Information technology – Open Systems Interconnection – Systems management: Metric objects and attributes.*
- ITU-T Recommendation X.741 (1995) | ISO/IEC 10164-9:1995, *Information technology – Open Systems Interconnection – Systems management: Objects and attributes for access control.*
- ITU-T Recommendation X.746 (1995) | ISO/IEC 10164-15:1995, *Information technology – Open Systems Interconnection – Systems management: Scheduling function.*

## 2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.209 (1988), *Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).*  
ISO/IEC 8825:1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*
- ITU-T Recommendation X.291 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Abstract test suite specification.*  
ISO/IEC 9646-2:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 2: Abstract Test Suite specification.*
- ITU-T Recommendation X.296 (1995), *OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications – Implementation conformance statement.*  
ISO/IEC 9646-7:1995, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements..*
- CCITT Recommendation X.700 (1992), *Management framework for Open Systems Interconnection (OSI) for CCITT Applications.*  
ISO/IEC 7498-4:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework.*

### 3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

#### 3.1 Basic Reference Model definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.200 | ISO/IEC 7498-1.

- a) open system;
- b) systems management.

#### 3.2 Service convention definitions

This Recommendation | International Standard makes use of the following term defined in ITU-T Rec. X.210 | ISO/IEC 10731.

- primitive.

#### 3.3 Management framework definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.700 | ISO/IEC 7498-4.

- a) management information;
- b) managed object.

#### 3.4 Systems management overview definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.701 | ISO/IEC 10040.

- a) agent role;
- b) management support object;
- c) managed object class;
- d) manager role;
- e) notification;
- f) systems management functional unit;
- g) system management operation.

#### 3.5 Common management information service definitions

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.710 | ISO/IEC 9595.

- a) attribute;
- b) common management information services.

#### 3.6 Additional definitions

The following terms are defined in this Recommendation | International Standard.

**3.6.1 command sequencer:** A management support object representing a resource which functions in a manager role as a notification destination and as an initiator of operations determined by its launch scripts, with the ability to delegate management activities.

**3.6.2 launch script:** A managed object representing the instructions to be performed by a command sequencer.

**3.6.3 thread:** A managed object representing the execution of a launch script. The execution results or errors from launch script executions are returned by the thread.

**3.6.4 suspendable thread:** The suspendable thread is derived from the thread managed object class. These threads are spawned by asynchronous launch pads. They can be suspended by means of suspend action directed at them and resumed by means of a resume action directed at them.

**3.6.5 launch pad:** A management support object to which a trigger may be directed to initiate the execution of a launch script. A launch pad serves as an Initial Value Managed Object (IVMO) for a thread.

**3.6.6 asynchronous launch pad:** An asynchronous launch pad is derived from launch pad. It returns a trigger result notification without waiting for results of execution of the launch scripts. Execution results or errors from launch script executions are notified directly from the thread.

**3.6.7 synchronous launch pad:** A synchronous launch pad is derived from the launch pad. It returns trigger result notification or processing error alarm after it gets all the execution results and errors from threads after the threads complete their execution.

**3.6.8 trigger activator:** An initiator of script execution by causing a launch pad to spawn one or more threads. It directs a command to a launch pad, in the form of scheduler, operations, notifications or local action.

**3.6.9 command:** An instruction for a management activity that is performed in the agent system in accordance with contents of a launch script. A command is described with a scripting language. Currently, the system management scripting language is defined in Annex F.

## 4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

|       |                                       |
|-------|---------------------------------------|
| ASN.1 | Abstract Syntax Notation One          |
| CMIS  | Common Management Information Service |
| CS    | Command Sequencer                     |
| IVMO  | Initial Value Managed Object          |
| OSI   | Open Systems Interconnection          |
| LP    | Launch Pad                            |
| SMSL  | Systems Management Scripting Language |

## 5 Conventions

This Specification defines services for the command sequencer following the descriptive conventions defined in ITU-T Rec. X.210 | ISO/IEC 10731. In clause 9, the definition of each service includes a table that lists the service parameters. For a given service primitive, the presence of each parameter is described by one of the following values:

|     |                                                                                                      |
|-----|------------------------------------------------------------------------------------------------------|
| M   | The parameter is mandatory.                                                                          |
| (=) | The value of the parameter is equal to the value of the parameter in the column to the left.         |
| U   | The use of the parameter is a Service-user option.                                                   |
| –   | The parameter is not present in the interaction described by the primitive concerned.                |
| C   | The parameter is conditional. The conditions are defined by the test which describes this parameter. |
| P   | The parameter is subject to the constraints imposed by ITU-T Rec. X.710   ISO/IEC 9595.              |

NOTE – The parameters which are marked “P” in service tables of this Specification are mapped directly onto the corresponding parameters of the CMIS service primitive, without changing the semantics or syntax of the parameters.

The font used for GDMO, ASN.1 and GRM in this Recommendation | International Standard is Courier. The BNF for SMSL in F.14.11 is in Courier New. In Annexes F and G, SMSL function parameters have been italicized.

## 6 Requirements

The requirements to be satisfied are:

- User requirements:
  - Allow the delegation of management activities.
  - Reduce the amount of communication that must occur between manager and agents.

- Allow delegated manager systems to operate on agent systems even when communications between a manager and the agent systems have been disrupted or are not possible.
- Provide flexible control of management activities.
- Provide a scripting language which can describe procedures to perform management operations.
- Allow delegated systems to execute CMIS operations in sequence.
- Operational requirements:
  - Pre-scheduled or delayed execution of a systems management operation.
  - Capabilities for modifying the request for pre-scheduled or delayed execution.
  - Capabilities for initiating, suspending, resuming and terminating systems management operations based on time management actions or the occurrence of events.
  - Capabilities for reporting and recording the outcome of pre-scheduled or delayed execution.
  - The ability to send notifications when state changes occur.

## 7 Model

### 7.1 Model description

The model describes how triggered, pre-scheduled or delayed execution of system management operations can be performed by the command sequencer. It describes the conceptual components, the relationship between these components, a description of the states and possible state transitions.

Figure 1 is a schematic description of the command sequencer capability of a system.

The functionality of a command sequencer is modeled by the launch script, thread, and launch pad objects. It is an OSI abstraction of pre-scheduled or delayed operation execution in open systems. A command sequencer may contain any number of launch pads, for which the command sequencer serves as a service provider. Each launch pad may execute one launch script at a time, or may execute multiple launch scripts at a time. On receiving a trigger from a trigger activator, a launch pad initiates the execution of a launch script. In addition to the trigger id component, the trigger may specify a launch script name (script id) and input arguments to the script as parameters within the execution parameter list component.

There are two types of launch pads, asynchronous launch pad and synchronous launch pad. An asynchronous launch pad returns a trigger result notification without waiting for results of execution of the launch scripts. Execution results or errors from launch script executions are notified directly from the thread. A synchronous launch pad, on the other hand, returns trigger result notification or processing error alarm after it gets all the execution results and errors from threads when the threads complete their execution.

A launch script instance may contain any number of individual instructions. The execution parameter list component of the trigger is a list of scripts (identified by their script ids) to be executed and the corresponding input parameters needed to execute those scripts. A default execution parameter list may be specified for a launch pad to execute in case it receives a trigger in which the trigger parameters are not specified. If the launch pad is not configured to execute a default execution parameter list and the execution parameter list component is not supplied by the trigger and if the launch pad receives a trigger attempting to activate it, a no script error code is returned in the error code field of the trigger result notification. The launch pad has an available script list attribute which can be configured to identify scripts that can be executed by it. If a execution parameter list component is present in the trigger, the launch pad verifies whether each script id from the execution parameter component is present in its available script list attribute. Only those script instances indicated by the script id which are present in the available script list attribute are executed. If none of the script ids are present in the available script list, the launch pad returns a script rejected error code in the trigger result, and script execution does not take place.

Specialized scripting language object classes are derived from the launch script object class. Hence these instructions may be specified as specialized script instances. Multiple sets of launch script instructions may be executed sequentially or in parallel by threads, in accordance with the execution parameter data type. Several nested levels of sub-threading may be necessary in order to execute script instances.

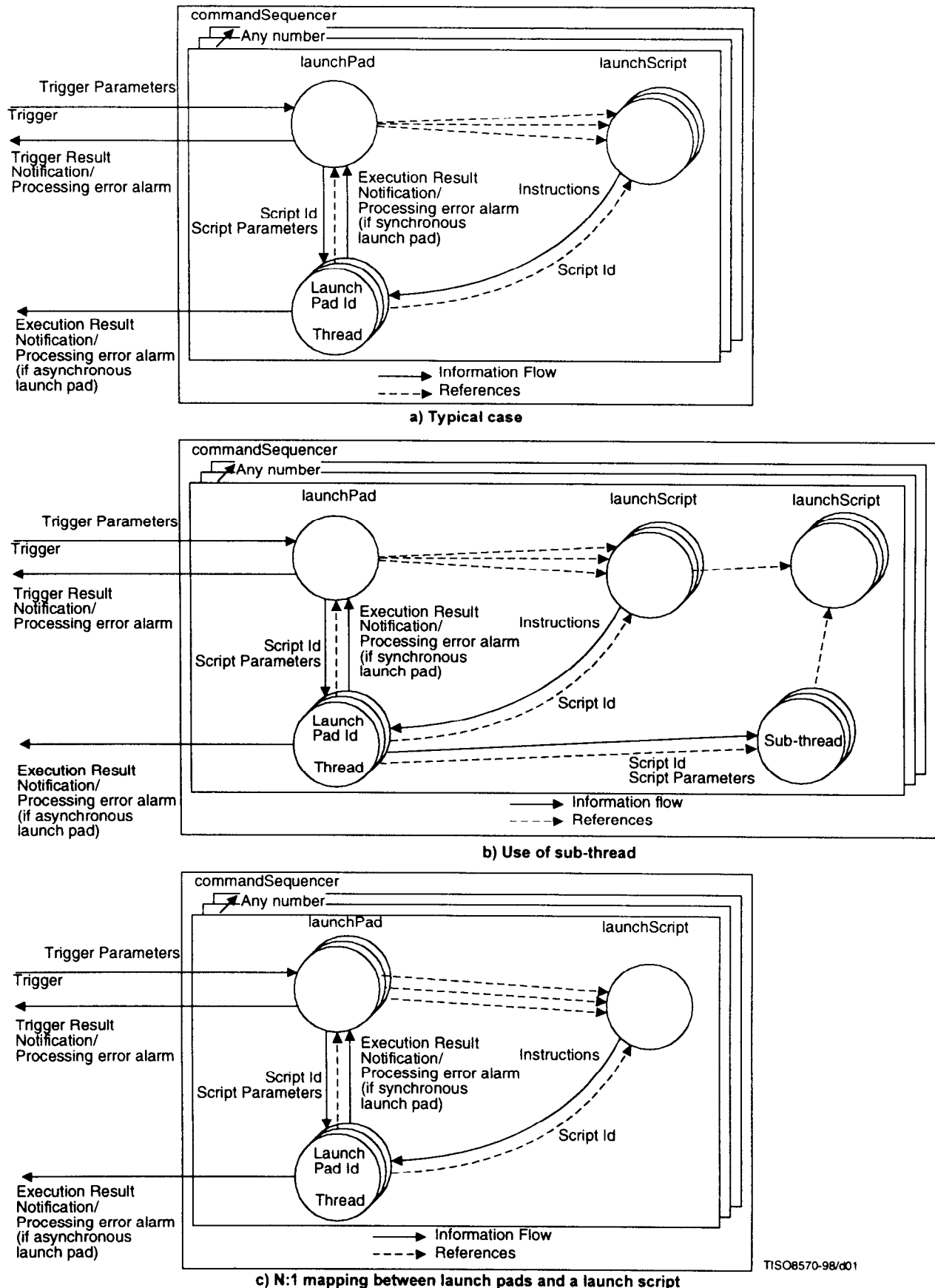


Figure 1 – Command sequencer model

In order to initiate the execution behaviour of launch script instances, a trigger shall be directed at the launch pad object instance. Unparameterized triggers may activate the launch pad in cases where the launch pad has a default execution parameter list.

The launch pad acts as an IVMO for a thread and supplies the execution parameter list to the executing parameters attribute of the thread. The execution parameter list may be a single execution parameter, a sequence of execution parameters or a set of execution parameters. The execution parameter is a sequence of script ids and script parameters. The script id identifies the managed object instance name of the scripting object instance to be executed and the script parameters supply the parameter values which are needed as inputs to the scripting object instance. If a sequence of execution parameters is specified, the launch pad spawns a thread in order to execute the script instances and supplies the script id and the script parameters (if required) from the execution parameters to the thread. On completion of the thread, this is repeated for the rest of the script ids in the list in sequence. If a set of execution parameters is specified, the launch pad supplies the set of script ids and script parameters (if required) to threads and the corresponding scripts instances are executed in parallel. The semantics of parameter passing between the launch pad and threads depend on the parameter passing mechanism supported by the scripting language in which the script is written.

One thread is assigned the execution of one script instance. This thread may spawn other threads if necessary. This may happen when one script instance invokes another script instances. When this happens, the thread executing a calling script spawns a sub-thread, passing the script id and script parameters (if required) of the called script to the sub-thread. The semantics of parameter passing between threads and sub-threads depend on the parameter passing mechanism supported by the scripting language in which the script is written.

Asynchronous launch pads should spawn suspendable threads. A suspendable thread can be suspended and resumed by means of suspend and resume actions respectively. Individual threads spawned by synchronous launch pads may not be suspended and resumed. All threads related to execution of a script may be suspended or resumed in both cases.

A thread is complete after all its sub-threads have completed successfully or reported an error. Once this happens, execution of a launch script is complete; the corresponding launch pad then returns to an inactive (idle) state. A thread is contained by the object which spawned it. A thread may be contained by a launch pad or another thread.

Multiple launch pads may reference a particular launch script. Multiple threads may reference that same launch script. The existence of a script is independent of any references to it by launch pads or threads. Launch scripts are defined in the specialized script classes derived from the launch script object class. The semantics and syntax of these scripts are specified in the definition of the scripting language in which the scripts are written. The scripting language definition also specifies a set of basic scripting functions which are necessary to provide control and processing ability to the launch scripts.

The general string script managed object class should be used for writing scripts which are represented in the form of a general string. The script language name attribute indicates the name of that scripting language and the script content attribute represents the script written in this scripting language in the form of a general string. Annexes F and G define a specialized scripting language, System Management Scripting Language (SMSL), as the scripting language in which scripts represented in the form of a general string should be written. It is possible to define other classes of scripts. Annex D defines, *cmisScript*, a scripting language, in the form of managed objects which can be used to write scripts in the CMIS environment.

## 7.2 Triggering process and reporting results

Triggers activators directed at the launch pad may be in various forms such as schedulers, operations, notifications and local action. When a launch pad receives a trigger, it spawns one or more threads in order to execute a script. After all threads related to a trigger are spawned, an asynchronous launch pad emits a trigger result notification which includes sets of thread id and script id. Results of script execution are propagated by threads as execution result notifications directly to the manager in the case of the asynchronous launch pad. After all threads related to a trigger are completed, a synchronous launch pad synchronizes all the execution results or errors from threads and emits a trigger result notification which includes sets of thread id, script id and execution results or errors to the manager.

The execution result type attribute of the script identifies the type of result expected from execution of the script and should correspond to the execution result type attribute of the execution result. The *errorCode* field of the execution result is set to the no error code when the execution is successful otherwise it is set to the appropriate error code.

An executing thread may terminate spontaneously either upon the completion of its execution or in abnormal conditions. In the latter case, the thread indicates abnormal termination by issuing a processing error alarm notification.

The execution result and processing error alarm notifications are issued by the thread and forwarded to appropriate notification destination(s). In the case of an asynchronous launch pad, these notifications are forwarded to external notification destinations whereas in the case of a synchronous launch pad, these notifications are propagated to the launch pad.

A manager may voluntarily terminate all launching processes by means of a delete operation to the corresponding launch pad. On receiving a delete operation, if the thread-launchPad name binding includes "DELETE DELETES-CONTAINED-OBJECTS" definition, all its threads which cause the execution of the script are terminated and deleted. The launch pad is then deleted.

A manager may voluntarily terminate all executions related to a thread by means of a delete operation to the corresponding thread. On receiving a delete operation, if the thread-thread name binding includes "DELETE DELETES-CONTAINED-OBJECTS" definition, all its sub-threads related to the execution of the script are terminated and deleted. The thread is then deleted.

In order to cause execution of all scripts being currently executed by a synchronous or asynchronous launch pad to be terminated, a terminate action may be directed at the launch pad. All threads related to the execution of scripts are terminated and deleted when a terminate action is received by the launch pad.

Launching of all threads being currently executed by a synchronous or asynchronous launch pad may be suspended by a suspend action directed at the launch pad and subsequently resumed by a resume action.

Execution of scripts by suspendable threads spawned by an asynchronous launch pad, may be suspended by a suspend action directed at the thread and subsequently resumed by a resume action. The thread id, returned in the trigger result notification should be supplied as a parameter to suspend and resume actions.

The launch pad has attributes to monitor a specific attribute of a specific object instance. If the value of the monitored attribute is changed, a trigger is generated to cause execution of a specified script list.

If the monitored attribute is an Event Discrimination Counter (EDC) counter as defined in Annex C, the notifications filtered by the EDC, trigger the launching of scripts by the launch pad.

7.3 Management of command sequencer

The attribute values of the launch pad, thread, launch script, and specialized scripting managed object instances are retrieved and modified through Get and Set operations, respectively.

Tables 1 to 5 map the status attributes of the command sequencer, launch pad, thread and script managed objects to the states defined in CCITT Rec. X.731 | ISO/IEC 10164-2.

NOTE – "-" means any value.

Table 1 – Status table of command sequencer

| Status of command sequencer | Administrative state | Operational state |
|-----------------------------|----------------------|-------------------|
| CS not operational          | –                    | disabled          |
| CS is operational           | unlocked             | enabled           |
| CS is locked                | locked               | enabled           |
| CS is shutting down         | shuttingDown         | enabled           |

When a command sequencer has a disabled operational state it is in a totally inoperable state and its launch pads are not executing scripts. If it is in an enabled state, an event which consists of an operation being performed at the managed object boundary may cause a transition from a locked administrative state to an unlocked state or vice versa. When the command sequencer goes into a locked state, it causes its launch pads to suspend the execution of launch scripts. Alternatively, when it goes into an unlocked administrative state, the launch pads are available to start or resume the execution of launch scripts.

**Table 2 – Status table of launch pad**

| Status of launch pad  | Administrative state | Operational state | Control status | Usage state | Availability status |
|-----------------------|----------------------|-------------------|----------------|-------------|---------------------|
| LP is not operational | –                    | disabled          | –              | –           | –                   |
| LP is operational     | unlocked             | enabled           | –              | Busy        | –                   |
| LP is operational     | unlocked             | enabled           | –              | Idle        | –                   |
| LP is locked          | locked               | enabled           | –              | Idle        | –                   |
| LP is on duty         | –                    | –                 | –              | –           | Not Off duty        |
| LP is off duty        | –                    | –                 | –              | –           | Off duty            |
| LP is suspended       | –                    | –                 | Suspended      | –           | –                   |
| LP is resumed         | –                    | –                 | Empty          | –           | –                   |

When a launch pad has a disabled operational state, it is in a totally inoperable state and cannot execute scripts. If it is in an enabled state, an event which consists of an operation being performed at the managed object boundary may cause it to transition from a locked administrative state to an unlocked state or vice versa. When the launch pad goes into a locked state, it suspends the execution of launch scripts. Alternatively, when it goes into an unlocked administrative state, the launch pads are available to start or resume the execution of launch scripts. The launch pad is made inactive by an internal control process according to a predetermined time schedule and its availability status value is off duty. A suspend action causes the control status to change to suspended and a resume action changes it back to its default value, empty.

**Table 3 – Status table of thread**

| Status of thread       | Operational state |
|------------------------|-------------------|
| Thread not operational | Disabled          |
| Thread operational     | Enabled           |

The thread is in an enabled state when it is performing a script execution and in a disabled state when it is not.

**Table 4 – Status table of suspendable thread**

| Status of suspendable thread       | Operational state | Control status |
|------------------------------------|-------------------|----------------|
| Suspendable thread not operational | Disabled          | –              |
| Suspendable thread operational     | Enabled           | –              |
| Suspendable thread suspended       | –                 | Suspended      |
| Suspendable thread resumed         | –                 | Empty          |

A suspend action causes the control status of the suspendable thread to change to suspended and a resume action changes it back to its default value, empty.

**Table 5 – Status table of launch script**

| Status of launch script  | Administrative state |
|--------------------------|----------------------|
| LS execution allowed     | Unlocked             |
| LS execution not allowed | Locked               |

An event which consists of an operation being performed at the managed object boundary may cause a script to transition from a locked administrative state to an unlocked state or vice versa. When the launch script goes into a locked state, it cannot be executed by a launch pad other than the ones which are currently executing it. Alternatively, when it goes into an unlocked administrative state, the launch script is available for execution by other launch pads.

## 7.4 Scheduling of the command sequencer

An external scheduler scheduling package provides the capability of scheduling the activation of a command sequencer's launch pads by triggers. The launch pad's availability status attribute will be changed to "off duty" or "not off duty" in accordance with the scheduling characteristics specified by an external scheduler managed object. The semantics of the external scheduler scheduling package are described in CCITT Rec. X.734 | ISO/IEC 10164-5 and CCITT Rec. X.735 | ISO/IEC 10164-6 .

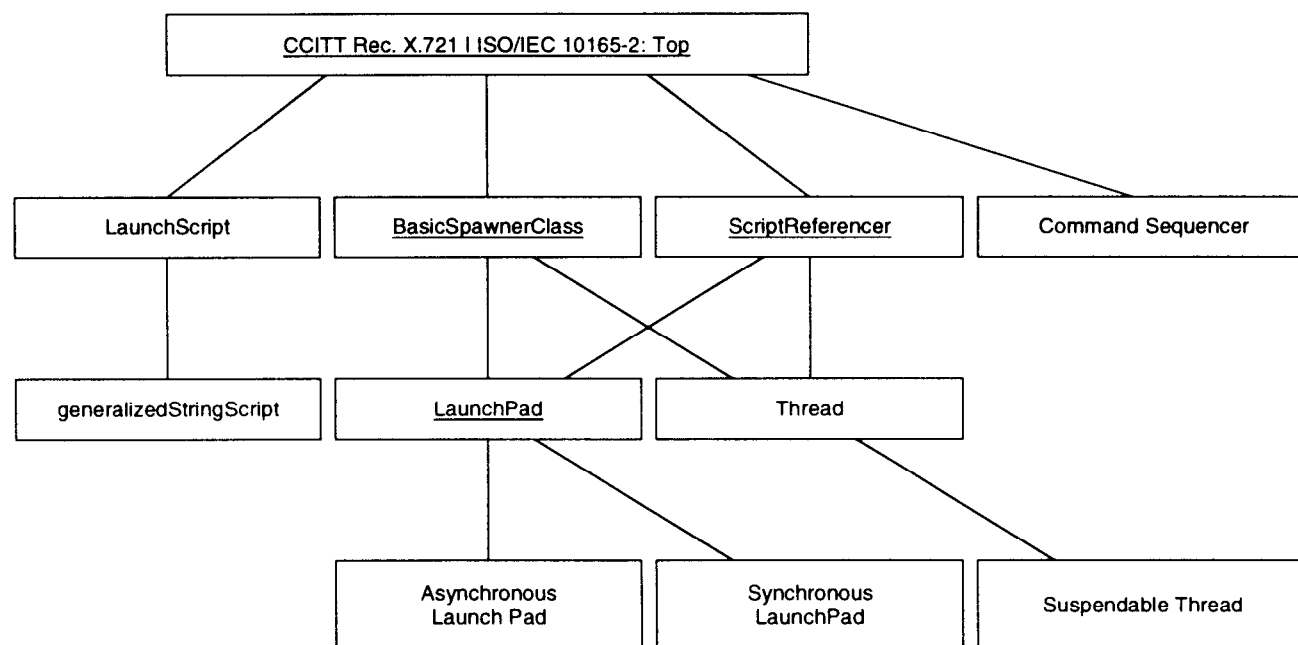
## 7.5 Access control

The command sequencer should be permitted access to all the managed object instances which are operated on by its threads. Behaviour of the thread when an operation on an instance is denied should be defined in the script. For example, if an operation is denied, an unauthorized access attempt error may be returned by a processing error alarm notification.

# 8 Generic definitions

## 8.1 Managed objects

This Specification defines a set of managed object classes. The inheritance structure of these managed object classes is shown in Figure 2.



TISO8580-98/d02

NOTE – Uninstantiable object classes are underlined.

**Figure 2 – Inheritance structure of resources of command sequencer**

The containment structure of these managed object classes is shown in Figure 3.

### 8.1.1 Command sequencer

#### 8.1.1.1 Overview

- A management support object representing a resource which acts in a manager role as an invoker of operations determined by its launch scripts and as a notification destination.
- Acts as service provider for a launch pad.
- Contains one or more launch pads.

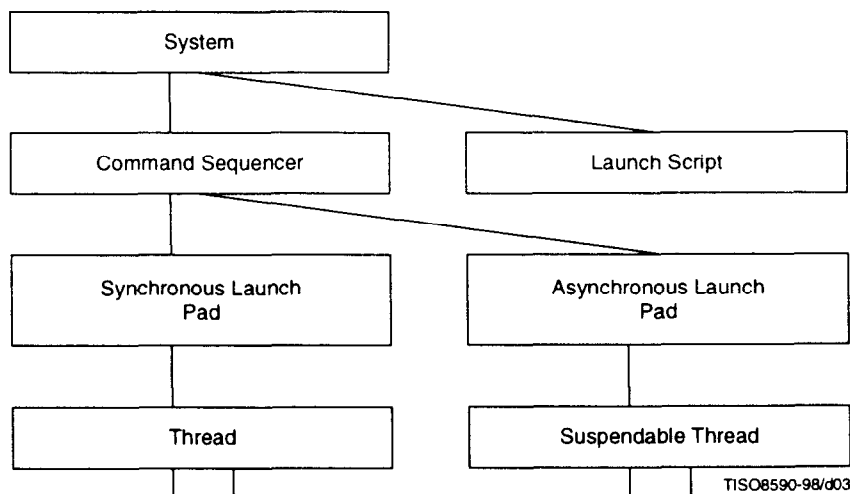


Figure 3 – Containment structure of resources of command sequencer

#### 8.1.1.2 Packages of the command sequencer management support object

The command sequencer management support object has the following mandatory package:

- command sequencer package.

#### 8.1.1.3 Characteristics of the command sequencer

The command sequencer management support object class has the following attributes.

##### 8.1.1.3.1 Command sequencer id

The value of this attribute defines an instance of the command sequencer management support object class.

##### 8.1.1.3.2 Administrative state

This attribute represents the administrative capability of the command sequencer to perform its function. The following administrative states are defined:

- a) Unlocked – The launch pads of the command sequencer are permitted to start or resume execution of launch scripts.
- b) Locked – The launch pads of the command sequencer are not permitted to start execution of launch scripts. If executions are in progress, they are suspended.
- c) Shutting down – The command sequencer is shutting down and its launch pads will not be permitted to perform any script executions.

##### 8.1.1.3.3 Operational state

This attribute represents the operational capability of the command sequencer to perform its functions.

The following operational states are defined:

- a) Enabled – The command sequencer is operational and ready for use.
- b) Disabled – The command sequencer is not available for use.

#### 8.1.1.4 Notifications of the command sequencer

The command sequencer support management object class has the following notifications:

- object creation, as defined in CCITT Rec. X.730 | ISO/IEC 10164-1.
- object deletion, as defined in CCITT Rec. X.730 | ISO/IEC 10164-1.
- state change, as defined in CCITT Rec. X.730 | ISO/IEC 10164-1.

## 8.1.2 Thread

### 8.1.2.1 Overview

- A management object that models command execution and performs commands according to a script.
- Subclass of the basic spawner and script referencer managed object classes.
- Each thread is dedicated to a synchronous launch pad.
- This object class has a notification which notifies the result of the execution of the launch script.

### 8.1.2.2 Packages of the thread managed object class

The thread managed object class has the following mandatory packages:

- thread package;
- execution result package.

### 8.1.2.3 Characteristics of the thread managed object class

The thread class has the following attributes.

#### 8.1.2.3.1 Thread Id

The value of this attribute identifies an instance of the thread management object class.

#### 8.1.2.3.2 Script Id

This value of this attribute identifies an instance of the launch script managed object class which is being executed.

#### 8.1.2.3.3 Executing parameters

The script parameters component of this attribute is a list of parameter values supplied by a launch pad, which it needs to perform a script execution. The script id component of this attribute indicates the script(s) that should be executed with the corresponding parameters.

#### 8.1.2.3.4 Operational state

This attribute represents the operational capability of the thread to perform its functions. The following operational states are defined:

- Enabled – The thread is operational and is performing a script execution.
- Disabled – The thread is not operational.

### 8.1.2.4 Notifications of the thread managed object class

The thread managed object class has the following notifications, which it forwards to the launch pad managed object class:

- execution result as defined in 8.2.1.
- processing error alarm as defined in CCITT X.733 | ISO/IEC 10164-4.

## 8.1.3 Suspendable thread

### 8.1.3.1 Overview

- Subclass of the thread managed object class.
- Spawned by an asynchronous launch pad.
- Can be suspended and resumed by means of suspend and resume actions.

### 8.1.3.2 Packages of the suspendable thread

The suspendable thread managed object class has the following mandatory package:

- suspend resume acceptor.

### 8.1.3.3 Characteristics of the suspendable thread

The suspendable thread has the following attributes.

**8.1.3.3.1 Control status**

The control status attribute is defined in CCITT Rec. X.731 | ISO/IEC 10164-2. The default value is empty. If execution is suspended, the control status changes to 'suspended' and if execution is resumed, the value changes to empty.

**8.1.3.4 Actions of the suspendable thread managed object class**

The following actions can be directed at the suspendable thread managed object class:

- suspend;
- resume.

**8.1.4 Launch script****8.1.4.1 Overview**

- Management information that represents a series of instructions in specialized scripting languages.
- Each script should be independent.

**8.1.4.2 Characteristics of the launch script**

The launch script class has the following attributes.

**8.1.4.2.1 Script Id**

The value of this attribute identifies an instance of the launch script managed object class.

**8.1.4.2.2 Execution result type**

The value of this attribute identifies the expected type of execution result.

**8.1.4.2.3 Administrative state**

This is as defined in CCITT Rec. X.731 | ISO/IEC 10164-2.

**8.1.4.3 Packages of the launch script object class**

The launch script managed object class has the following mandatory package:

- launch script package.

**8.1.5 Basic spawner class****8.1.5.1 Overview**

- Signifies capability of creation of new contained object instances with automatic name generation for these new instances.
- Superclass of command sequence thread and launch pad object classes.

**8.1.5.2 Packages of the basic spawner class**

The basic spawner class has the following mandatory package:

- basic spawner package.

**8.1.6 Launch pad****8.1.6.1 Overview**

- Subclass of the basic spawner and script referencer managed object classes.
- Initiator of launch script execution on receiving a trigger.
- Acts as IVMO for a thread.
- A script is executed by the launch pad by means of one or more threads.

### 8.1.6.2 Packages of the launch pad managed object class

The mandatory packages of the launch pad managed object class are:

- launch pad package;
- trigger action acceptor;
- parameter passer;
- trigger result package;
- trigger event acceptor;
- terminate acceptor;
- external scheduler;
- suspend resume acceptor.

### 8.1.6.3 Characteristics of launch pad

The launch pad managed object class has the following attributes.

#### 8.1.6.3.1 Available script list

This is a list of all the scripts which a launch pad is capable of executing. The launch pad executes only those scripts which are present both in the execution parameter list component of trigger parameters of the trigger and in the available script list attribute.

#### 8.1.6.3.2 Default execution parameter list

This is a list of script ids and script parameters which are used for default execution when a launch pad is triggered with no parameters.

#### 8.1.6.3.3 Administrative state

This attribute represents the administrative capability of the launch pad to perform its function. The following administrative states are defined:

- a) Unlocked – The launch pad is permitted to start or resume execution of launch scripts.
- b) Locked – The launch pad is not permitted to start execution of launch scripts. If executions are in progress, they are suspended.

#### 8.1.6.3.4 Operational state

This attribute represents the operational capability of the launch pad to perform its functions.

The following operational states are defined:

- a) Enabled – The launch pad is operational and is available to execute a script.
- b) Disabled – The launch pad is not operational and is unavailable for script executions.

#### 8.1.6.3.5 Usage state

This attribute represents the usage status of the launch pad. The following usage states are defined:

- a) Busy – The launch pad is being used to execute a launch script.
- b) Idle – The launch pad is not being used to execute a script.

#### 8.1.6.3.6 Availability status

This status condition indicates whether the launch pad is available to perform its function. The following states are defined:

- a) Off Duty  
The launch pad had been made inactive by an internal control process in accordance with a predetermined time schedule.
- b) Not Off Duty  
The availability status attribute does not have the Off Duty value and hence has been made active.

**8.1.6.3.7 Observed object instance**

This is defined in ITU-T Rec. X.739 | ISO/IEC 10164-11.

**8.1.6.3.8 Observed attribute id**

This is defined in ITU-T Rec. X.739 | ISO/IEC 10164-11.

**8.1.6.3.9 Control status**

This attribute is defined in CCITT Rec. X.731 | ISO/IEC 10164-2. The default value is empty. If execution is suspended, the control status changes to 'suspended' and if execution is resumed, the value changes to empty.

**8.1.6.3.10 Launch pad id**

This attribute names an instance of the launch pad managed object class.

**8.1.6.4 Notifications of the launch pad managed object class**

The launch pad managed object class has the following notifications, which can be forwarded to the appropriate notification destination(s):

- trigger result as defined in 8.2.1.
- processing error alarm as defined in CCITT X.733 | ISO/IEC 10164-4.

**8.1.7 Asynchronous launch pad****8.1.7.1 Overview**

- Subclass of the launch pad managed object class.
- Suspendable threads are spawned by asynchronous launch pad.
- Sends out a trigger result notification as soon as all suspendable threads are spawned.

**8.1.7.2 Packages of the asynchronous launch pad****8.1.7.3 Managed object class**

The asynchronous launch pad managed object class has the following package:

- triggerAsynchronousResultPackage.

**8.1.8 Synchronous launch pad****8.1.8.1 Overview**

- Subclass of the launch pad managed object class.
- Threads are spawned by the synchronous launch pad.
- Sends out a trigger result notification after synchronizing results as soon as all threads are complete.

**8.1.8.2 Packages of the synchronous launch pad managed object class**

The asynchronous launch pad managed object class has the following package:

- trigger synchronous result package.

**8.1.9 General string script****8.1.9.1 Overview**

- Subclass of the launch script object class.
- Scripts which can be represented in the form of a general string.
- It is possible to add other specialized scripts as subclasses.

**8.1.9.2 Characteristics of general string scripting language**

The general string scripting language managed object class has the following attributes.

**8.1.9.2.1 Scripting language name**

This is the name of the language which defines the syntactic and semantic properties of a script which is represented as a general string.

**8.1.9.2.2 script content**

This point to the script which is represented as a general string.

**8.1.9.3 Packages of general string scripting language**

The general string scripting language managed object class has the following mandatory package:

- general string script package.

**8.1.10 Script referencer****8.1.10.1 Overview**

- Superclass for launch pad and thread managed object classes.
- Defines a reference relationship mapping between launch pad and launch script and thread and launch script.

**8.1.10.2 Packages of script referencer**

The script referencer has the following mandatory package:

- script referencer package.

**8.2 Generic notifications**

The following notifications are defined in this Specification.

**8.2.1 Trigger result**

This returns the result of a script execution and with the value of errorCode set to noError if it was successful or an appropriate error code to indicate the nature of failure. The error code which can be returned in the errorCode field of the executionResult notification are:

- no error, if execution was successful;
- no script error, if execution failed because script name was not specified;
- script rejected error, if a script was not in the list of scripts which a launch pad is configured to execute;
- invalid parameter type error, if there is a type mismatch between the parameter type expected by the script and that supplied by the script;
- invalid parameter value error, if the value supplied in the parameter is invalid, e.g. out of range;
- script syntax error, if the script execution failed due to a syntax error in the script;
- script execution failed error, if the script execution failed for reasons other than improper syntax;
- invalid parameter number, if the number of parameters supplied are inconsistent with the number of parameters expected by the script;
- unauthorized access error, if access for one or more object instances to be used by the script, is denied.

**8.2.2 Execution result**

This returns the result of a script execution and with the value of errorCode set to noError if it was successful or an appropriate error code to indicate the nature of failure. The error code which can be returned in the errorCode field of the executionResult notification are:

- no error, if execution was successful;
- no script error, if execution failed because script name was not specified;
- script rejected error, if a script was not in the list of scripts which a launch pad is configured to execute;
- invalid parameter type error, if there is a type mismatch between the parameter type expected by the script and that supplied by the script;
- invalid parameter value error, if the value supplied parameter is invalid, e.g. out of range;
- script syntax error, if the script execution failed due to a syntax error in the script;

- script execution failed error, if the script execution failed for reasons other than improper syntax;
- invalid parameter number, if the number of parameters supplied are inconsistent with the number of parameters expected by the script;
- unauthorized access error, if access for one or more object instances to be used by the script, is denied.

### 8.3 Generic actions

The following action types are defined within this Specification. These actions have been defined for the launch pad and the suspendable thread managed object classes in this Specification.

#### 8.3.1 Suspend action

The suspend action directed at a managed object causes that object to suspend execution of the script that it is currently executing. The parameters carried by the suspend action are the trigger id to identify this action and either the thread id if the action is directed at a thread or the launch pad id, if this action is directed at a launch pad.

#### 8.3.2 Resume action

The resume action directed at a managed object causes that object to resume execution of the script which has been suspended by a previous suspend action. The parameters carried by the suspend action are the trigger id to identify this action and either the thread id if the action is directed at a thread or the launch pad id, if this action is directed at a launch pad.

#### 8.3.3 Terminate action

The terminate action directed at a managed object causes unconditional termination of any script executions by that object and any objects spawned by this object. The trigger id parameter is used to identify this action.

#### 8.3.4 Trigger action

A trigger action causes initiation of script execution. It supplies the trigger id parameter which identifies this action and its execution parameter list component consists of a sequence of script ids which indicate the scripts which should be executed and script parameters which are needed in order to execute these scripts.

## 9 Services

### 9.1 Introduction

The command sequencer provides services to modify the operations of command sequencer and launch scripts. In particular, the operations that can be applied to each instance of a command sequencer and launch scripts are:

- creation of command sequencer, launch pad and launch script instances;
- deletion of launch pad and launch script instances;
- modification of command sequencer, launch pad and launch script attributes;
- retrieval of command sequencer, launch pad and launch script attributes.

In addition to the above services to modify instances, this function provides notification services and action to trigger the command execution.

### 9.2 Initiation, Termination, Modification and Retrieval Services

The PT-CREATE, PT-DELETE, PT-SET and PT-GET services may be used to create, delete, modify and retrieve attribute values of command sequencer management support object and launch pad and launch script managed object instances.

### 9.3 Notification services

#### 9.3.1 Execution result service definition

This subclause specifies the executionResultInfo report service which is defined in this Recommendation | International Standard and maps it to the CMIS M-EVENT-REPORT service.

Table 6 – Execution result reporting parameters

| Parameter name          | Req/Ind | Rsp/Cnf |
|-------------------------|---------|---------|
| Invoke identifier       | P       | P       |
| Mode                    | P       | –       |
| Managed object class    | P       | P       |
| Managed object instance | P       | P       |
| Event type              | M       | C (=)   |
| Event time              | P       | –       |
| Event information       |         |         |
| trigger id              | M       | –       |
| script id               | M       | –       |
| thread id               | M       | –       |
| error code              | U       | –       |
| execution result type   | U       | –       |
| execution result        | U       | –       |
| Current time            | –       | P       |
| Event reply             | –       | P       |
| Errors                  | –       | P       |

Table 7 – Trigger result reporting parameters

| Parameter name          | Req/Ind | Rsp/Cnf |
|-------------------------|---------|---------|
| Invoke identifier       | P       | P       |
| Mode                    | P       | –       |
| Managed object class    | P       | P       |
| Managed object instance | P       | P       |
| Event type              | M       | C (=)   |
| Event time              | P       | –       |
| Event information       |         |         |
| trigger id              | M       | –       |
| script id               | U       | –       |
| thread id               | –       | –       |
| script parameters       | U       | –       |
| execution type          | U       | –       |
| error code              | U       | –       |
| execution result        | U       | –       |
| Current time            | –       | P       |
| Event reply             | –       | P       |
| Errors                  | –       | P       |

## 9.4 Action services

This subclause specifies the trigger and terminate action services which are defined in this Recommendation 1 International Standard and maps them to the CMIS M-EVENT-ACTION service.

**Table 8 – Trigger action service parameters**

| Parameter name          | Req/Ind | Rsp/Conf |
|-------------------------|---------|----------|
| Invoke identifier       | P       | P        |
| Linked identifier       | –       | P        |
| Mode                    | P       | –        |
| Base object class       | P       | –        |
| Base object instance    | P       | –        |
| Scope                   | P       | –        |
| Filter                  | P       | –        |
| Managed object class    | –       | P        |
| Managed object instance | –       | P        |
| Access control          | P       | –        |
| Synchronization         | P       | –        |
| Action type             | M       | –        |
| Action Information      |         |          |
| trigger id              | M       | –        |
| Attribute list          | U       | –        |
| Errors                  | –       | P        |

**Table 9 – Terminate action service parameters**

| Parameter name          | Req/Ind | Rsp/Conf |
|-------------------------|---------|----------|
| Invoke identifier       | P       | P        |
| Linked identifier       | –       | P        |
| Mode                    | P       | –        |
| Base object class       | P       | –        |
| Base object instance    | P       | –        |
| Scope                   | P       | –        |
| Filter                  | P       | –        |
| Managed object class    | –       | P        |
| Managed object instance | –       | P        |
| Access control          | P       | –        |
| Synchronization         | P       | –        |
| Action type             | M       | –        |
| Action Information      |         |          |
| trigger id              | M       | –        |
| Errors                  | –       | P        |

Table 10 – Suspend action service parameters

| Parameter name          | Req/Ind | Rsp/Conf |
|-------------------------|---------|----------|
| Invoke identifier       | P       | P        |
| Linked identifier       | –       | P        |
| Mode                    | P       | –        |
| Base object class       | P       | –        |
| Base object instance    | P       | –        |
| Scope                   | P       | –        |
| Filter                  | P       | –        |
| Managed object class    | –       | P        |
| Managed object instance | –       | P        |
| Access control          | P       | –        |
| Synchronization         | P       | –        |
| Action type             | M       | –        |
| Action Information      |         |          |
| trigger id              | M       | –        |
| thread id               | U       | –        |
| launch pad id           | U       | –        |
| Errors                  | –       | P        |

Table 11 – Resume action service parameters

| Parameter name          | Req/Ind | Rsp/Conf |
|-------------------------|---------|----------|
| Invoke identifier       | P       | P        |
| Linked identifier       | –       | P        |
| Mode                    | P       | –        |
| Base object class       | P       | –        |
| Base object instance    | P       | –        |
| Scope                   | P       | –        |
| Filter                  | P       | –        |
| Managed object class    | –       | P        |
| Managed object instance | –       | P        |
| Access control          | P       | –        |
| Synchronization         | P       | –        |
| Action type             | M       | –        |
| Action Information      |         |          |
| trigger id              | M       | –        |
| thread id               | U       | –        |
| launch pad id           | U       | –        |
| Errors                  | –       | P        |

## 10 Functional units

Three functional unit are defined in this Recommendation | International Standard for the management of command sequencers:

- a) *Execution functional unit*  
The execution functional unit requires the services of PT-CREATE, trigger action execution result notification service and processing error alarm reporting service.
- b) *Monitoring functional unit*  
The monitoring functional unit requires the services of PT-GET.
- c) *Control functional unit*  
The control functional unit requires the services of terminate action, PT-DELETE.

## 11 Protocols and abstract syntax

### 11.1 Abstract syntax

#### 11.1.1 Managed objects

##### 11.1.1.1 Defined managed objects

Table 12 identifies the relationship between the managed objects defined in 8.1 and the managed object class specification in Annex A.

**Table 12 – Managed objects and reference labels**

| Managed object name     | Reference label       |
|-------------------------|-----------------------|
| Basic spawner           | basicSpawnerClass     |
| Suspendable thread      | suspendableThread     |
| Thread                  | thread                |
| Asynchronous launch pad | asynchronousLaunchPad |
| Synchronous launch pad  | synchronousLaunchPad  |
| Launch pad              | launchPad             |
| Launch script           | launchScript          |
| Command sequencer       | commandSequencer      |
| General string script   | generalStringScript   |
| Script Referencer       | scriptReferencer      |

### 11.2 Attributes

#### 11.2.1 Attributes imported from the definition of management information

This Specification references the following management attributes, whose abstract syntax are specified in CCITT Rec. X.721 | ISO/IEC 10165-2:

- a) administrativeState;
- b) usageState;
- c) operationalState;
- d) controlStatus;
- e) availabilityStatus.

It also references the following management attributes, whose abstract syntax are specified in ITU-T Rec. X.739 | ISO/IEC 10164-11:

- a) observedObjectInstance;
- b) observedAttributeId.

### 11.2.2 Attributes defined in this Specification

This Specification defines the following management attributes, whose abstract syntax are specified in Annex A:

- a) launchPadId;
- b) commandSequencerId;
- c) threadId;
- d) scriptId;
- e) triggerId;
- f) executionResultType;
- g) executingParameters;
- h) scriptLanguageName;
- i) scriptContent;
- j) defaultExecutionParameterList;
- k) availableScriptList.

### 11.2.3 Parameter to Attribute Mapping

Table 13 identifies the relationship between the service parameters defined in 8.1 and 8.2 and the attribute type specifications in Annex A.

**Table 13 – Parameters and attribute names**

| Parameter name                   | Attribute name                |
|----------------------------------|-------------------------------|
| Launch pad id                    | launchPadId                   |
| Command sequencer id             | commandSequencerId            |
| Thread id                        | threadId                      |
| Script id                        | scriptId                      |
| Trigger id                       | triggerId                     |
| Execution result type            | executionResultType           |
| Executing parameters             | executingParameters           |
| Script language name             | scriptLanguageName            |
| Script content                   | scriptContent                 |
| Default execution parameter list | defaultExecutionParameterList |
| Available script list            | availableScriptList           |

## 11.4 Notifications

### 11.4.1 Referenced notifications

This Specification references the following events defined in CCITT Rec. X.730 | ISO/IEC 10164-1:

- a) object creation notification;
- b) object deletion notification;
- c) processing error alarm notification.

This Specification also references the following events defined in CCITT Rec. X.731 | ISO/IEC 10164-2:

- state change notification.

### 11.4.2 Notifications defined in this Specification

Table 14 identifies the relationship between the notifications defined in 9.3 and the notification type specifications in Annex A.

Table 14 – Notifications

| Event type       | Notification type   |
|------------------|---------------------|
| Trigger result   | triggerResultInfo   |
| Execution result | executionResultInfo |

## 11.5 Actions

### 11.5.1 Actions defined in this Specification

Table 15 identifies the relationship between the actions defined in 9.4 and the notification type specifications in Annex A.

Table 15 – Actions

| Action name | Reference label |
|-------------|-----------------|
| terminate   | terminate       |
| suspend     | suspend         |
| resume      | resume          |
| trigger     | trigger         |

## 11.6 Negotiation of functional units

This Recommendation | International Standard assigns the following object identifier value:

**{joint-iso-itu-t ms(9) function(2) part21(21) functionalUnitPackage(1)}**

as a value of the ASN.1 type FunctionalUnitPackageId defined in CCITT Rec. X.701 | ISO/IEC 10040 to use for negotiating the following functional units:

- 0 Execution functional unit
- M Monitoring functional unit
- 2 Control functional unit

where the number identifies the bit positions in the BIT STRING assigned to the functional units, and the names referencing the functional units are defined in clause 10.

Within the Systems management application context, the mechanism for negotiating the functional units is described by CCITT Rec. X.701 | ISO/IEC 10040.

NOTE – The requirement to negotiate functional units is specified by the application context.

## 12 Relationship with other functions

The command sequencer uses the services defined in CCITT Rec. X.731 | ISO/IEC 10164-2 for the notification of state changes, the services defined in CCITT Rec. X.730 | ISO/IEC 10164-1 for the creation and deletion of managed objects, the retrieval of attributes and notification of attribute value changes.

The command sequencer uses the services defined in ITU-T Rec. X.741 | ISO/IEC 10164-9 to provide access control capabilities to managed object instances which can be operated upon by threads.

## 13 Conformance

There are two conformance classes: general conformance class and dependent conformance class. A system claiming to implement the elements of procedure for systems management services referenced by this Specification shall comply with the requirements for either the general or the dependent conformance class as defined in the following subclauses. The supplier of the implementation shall state the class to which the conformance is claimed.

### 13.1 General conformance class requirements

A system claiming general conformance shall support this function for all managed object classes that import the management information defined in this Specification.

NOTE – This is applicable to all subclasses of the management support object classes defined in this Specification.

#### 13.1.1 Static conformance

The system shall:

- a) support the role of manager or agent or both, with respect to the control metrics functional unit and the monitor metrics functional unit;
- b) support the transfer syntax derived from the encoding rules specified in CCITT Rec. X.209 | ISO/IEC 8825 and named {joint-iso-itu-t asn1(1) basicEncoding(1)}, for the purpose of generating and interpreting the MAPDUs, defined by the abstract data types referenced in 11.4 and 11.5.
- c) when acting in the agent role, support one or more instances of at least one of the command sequencer, launch pad, launch script, thread managed object classes or any of their subclasses.

#### 13.1.2 Dynamic conformance

The system shall, in the role(s) for which conformance is claimed:

- a) Support the elements of procedure defined in:
  - CCITT Rec. X.730 | ISO/IEC 10164-1 for the PT-GET, PT-CREATE, PT-DELETE, PT-SET, object creation reporting, object deletion reporting and attribute change reporting services;
  - CCITT Rec. X.731 | ISO/IEC 10164-2 for the state change reporting service.
  - CCITT Rec. X.733 | ISO/IEC 10164-4 for the processing error alarm reporting service.
- b) Support the elements of procedure defined in this Specification for the following reporting and action services:
  - execution result notification;
  - trigger action;
  - terminate action;
  - suspend action;
  - resume action.

### 13.2 Dependent conformance class requirements

#### 13.2.1 Static conformance

The system shall:

- a) support the transfer syntax derived from the encoding rules specified in CCITT Rec. X.209 | ISO/IEC 8825 and named {joint-iso-itu-t asn1(1) basicEncoding(1)}, for the purpose of generating and interpreting the MAPDUs, defined by the abstract data types referenced in 11.1, as required by a referencing specification;
- b) support one or more instances of one of the command sequencer, launch pad, launch script, thread managed object classes or any of their subclasses, when acting in the agent role.

#### 13.2.2 Dynamic conformance

The system shall support the elements of procedure referenced by this Specification, as required by a referencing specification.

### 13.3 Conformance to support managed object definitions

The command sequencer objects supported by the open system shall comply with the behaviour specified in clause 8 and the syntax specified in Annex A.

## Annex A

### Definition of Management Information

(This annex forms an integral part of this Recommendation | International Standard)

#### A.1 Managed object class definitions

##### A.1.1 Basic objects

basicSpawnerClass MANAGED OBJECT CLASS

DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":top;

CHARACTERIZED BY basicSpawnerPackage ;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx1(1)};

commandSequencer MANAGED OBJECT CLASS

DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":top;

CHARACTERIZED BY

commandSequencerPackage PACKAGE

BEHAVIOUR commandSequencerBehaviour BEHAVIOUR

DEFINED AS "An instance of this class represents a resource acting in a manager role as an invoker of operations determined by its launch scripts.";;

ATTRIBUTES

commandSequencerId GET,

"CCITT Rec. X.731|ISO/IEC 10164-2:1992": administrativeState GET-REPLACE,

"CCITT Rec. X.731|ISO/IEC 10164-2:1992":operationalState GET;

NOTIFICATIONS

"CCITT Rec. X.730 | ISO/IEC 10164-1": objectCreation,

"CCITT Rec. X.730 | ISO/IEC 10164-1": objectDeletion,

"CCITT Rec. X.731 | ISO/IEC 10164-2": stateChange;;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21)

managedObjectClass(3) xx2(2)};

generalStringScript MANAGED OBJECT CLASS

DERIVED FROM launchScript;

CHARACTERIZED BY generalStringScriptPackage;;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx3(3)};

asynchronousLaunchPad MANAGED OBJECT CLASS

DERIVED FROM launchPad;

CHARACTERIZED BY triggerAsynchronousResultPackage;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx4(4)};

synchronousLaunchPad MANAGED OBJECT CLASS

DERIVED FROM launchPad;

CHARACTERIZED BY triggerSynchronousResultPackage;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx5(5)};

## ISO/IEC 10164-21 : 1998 (E)

```
launchPad    MANAGED OBJECT CLASS
DERIVED FROM basicSpawnerClass, scriptReferencer;
CHARACTERIZED BY
    launchPadPackage,
        triggerActionAcceptor,
        parameterPasser,
        triggerResultPackage,
        triggerEventAcceptor,
        terminateAcceptor,
        "CCITT Rec. 721 | ISO/IEC 10165-2:1992": externalScheduler,
        suspendResumeAcceptor;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx6(6)};

launchScript MANAGED OBJECT CLASS
DERIVED FROM "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":top;
CHARACTERIZED BY
    launchScriptPackage PACKAGE
    BEHAVIOUR launchScriptBehaviour BEHAVIOUR
    DEFINED AS "This managed object represents instructions to be carried out by a
    command sequencer.";;
    ATTRIBUTES
        scriptId GET,
        executionResultType GET,
        "CCITT Rec. X.721 | ISO /IEC 10165-2:1992": administrativeState GET-
        REPLACE;;;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx7(7)};

--
-- The following non-instantiable superclass simplifies the description of the
-- relationship between a launch pad and its scripts, along with the description
-- of the relationship between threads and scripts. Both the launch pad and
-- thread classes include it in their inheritance hierarchies.
--

scriptReferencer MANAGED OBJECT CLASS
DERIVED FROM "ITU-T Rec. X.725 | ISO/IEC 10165-7": genericRelationshipObject;
CHARACTERIZED BY scriptReferencerPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx8(8)};

thread MANAGED OBJECT CLASS
    DERIVED FROM basicSpawnerClass, scriptReferencer;
    CHARACTERIZED BY threadPackage, executionResultPackage;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3) xx9(9)};

suspendableThread MANAGED OBJECT CLASS
    DERIVED FROM thread;
    CHARACTERIZED BY suspendResumeAcceptor;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) managedObjectClass(3)
xx10(10)};
```

## A.2 Package definitions

### A.2.1 Basic packages

```

basicSpawnerPackage  PACKAGE
    BEHAVIOUR spawnerBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx1(1)};

generalStringScriptPackage  PACKAGE
    BEHAVIOUR generalStringScriptBehaviour;
    ATTRIBUTES scriptLanguageName GET-REPLACE,
               scriptContent GET-REPLACE;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx2(2)};

parameterPasser  PACKAGE
    BEHAVIOUR
        parameterPasserBehaviour;
    ATTRIBUTES
        "CCITT Rec. X.721 | ISO /IEC 10165-2:1992": administrativeState,
        "CCITT Rec. X.721 | ISO /IEC 10165-2:1992": operationalState,
        "CCITT Rec. X.721 | ISO /IEC 10165-2:1992": usageState,
        "CCITT Rec. X.721 | ISO /IEC 10165-2:1992": availabilityStatus;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx3(3)};

executionResultPackage  PACKAGE
    BEHAVIOUR executionResultBehaviour;;
    NOTIFICATION executionResultInfo;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx4(4)};

launchPadPackage  PACKAGE
    BEHAVIOUR launchPadBehaviour;
    ATTRIBUTES launchPadId GET;
    NOTIFICATIONS
        "CCITT Rec. X.721 | ISO/IEC 10165-2:1992": processingErrorAlarm;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx5(5)};

scriptReferencerPackage  PACKAGE
    BEHAVIOUR scriptReferencerBehaviour;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx6(6)};

suspendResumeAcceptor  PACKAGE
    BEHAVIOUR suspendResumeBehaviour;
    ATTRIBUTES "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":controlStatus GET;
    ACTIONS suspend, resume;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx7(7)};

```

terminateAcceptor PACKAGE

ACTIONS terminate;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx8(8)};

threadPackage PACKAGE

BEHAVIOUR threadBehaviour,

simpleScriptExecutionBehaviour;

ATTRIBUTES scriptId GET,

threadId GET,

executingParameters GET SET-BY-CREATE,

"CCITT Rec. X.731|ISO/IEC 10164-2:1992": operationalState GET;

NOTIFICATIONS

"CCITT Rec. X.734|ISO/IEC 10164-5": processingErrorAlarm;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx9(9)};

triggerActionAcceptor PACKAGE

BEHAVIOUR spawnerBehaviour,

triggerActionAcceptorBehaviour;;

ATTRIBUTES defaultExecutionParameterList REPLACE WITH DEFAULT

GET-REPLACE

SET BY CREATE

DEFAULT VALUE CSModule.emptyExecutionParameterList;

availableScriptList REPLACE WITH DEFAULT

ADD-REMOVE

GET-REPLACE

SET BY CREATE

DEFAULT VALUE CSModule.emptyScriptList;

ACTIONS Trigger;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx10(10)};

triggerEventAcceptor PACKAGE

BEHAVIOUR triggerEventAcceptorBehaviour;

ATTRIBUTES

REPLACE, "ITU-T Rec. X.739 (1993)|ISO/IEC 10164-11:1994": observedObjectInstance GET-

REPLACE; "ITU-T Rec. X.739 (1993)|ISO/IEC 10164-11:1994": observedAttributeId GET-

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx11(11)};

triggerAsynchronousResultPackage PACKAGE

BEHAVIOUR triggerAsynchronousResultBehaviour;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx12(12)};

triggerSynchronousResultPackage PACKAGE

BEHAVIOUR triggerSynchronousResultBehaviour;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx13(13)};

```
triggerResultPackage PACKAGE
    BEHAVIOUR    triggerResultBehaviour;;
    NOTIFICATION triggerResultInfo;
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx14(14)};
```

### A.3 Behaviour definitions

spawnerBehaviour BEHAVIOUR

DEFINED AS "Instances of this class are capable of causing the creation of new object instances. The newly created instances will be contained by this instance, and their names will be automatically generated. Until all created objects are complete, this object's usage status will be "in use". If this object's administrative state is "locked" such new objects cannot be created. If, due to local resource limitations, this object is incapable of supporting more contained objects, its usage status will be "busy".

When an instance of this class causes the creation of new objects, it serves as an IVMO during the creation by supplying values for the new object's attributes based on its own defaultExecutionParameterList attribute or any parameters which were supplied to it as part of the action or local mechanism which triggered the spawning of the new instance.

An instance of this class may cause the creation of new object instances from a single script id, from a set of script ids in any order or from a sequence of script ids in the order specified in the list, i.e. after the first has been created the second may not be created until after the first is completed, and so on. The value of the created object's scriptId attribute gets its value from the corresponding element of this object's script list.!!

executionResultBehaviour BEHAVIOUR

DEFINED AS "Instances of a class supporting this behaviour report intermediate and final results from execution of a thread.";

triggerAsynchronousResultBehaviour BEHAVIOUR

DEFINED AS "As soon as all threads that must be launched by one trigger is launched, the launch pad issues the triggerResultInfo notification.";

triggerSynchronousResultBehaviour BEHAVIOUR

DEFINED AS "As soon as all threads that must be launched by one trigger have completed, the launch pad issues the triggerResultInfo notification which contains execution results or errors.";

triggerResultBehaviour BEHAVIOUR

DEFINED AS "The launch pad issues the triggerResultInfo notification. ";

scriptReferencerBehaviour BEHAVIOUR

DEFINED AS "A script referencer is a non-instantiable object class which defines a reference relationship mapping from instances of the launch pad and the launch script managed object classes and from instances of the thread and the launch script managed object classes.";

suspendResumeBehaviour BEHAVIOUR

DEFINED AS "Execution of a script by a thread may be suspended by a suspend action directed at the thread or launch pad and subsequently resumed by a resume action. Default value of controlStatus is empty. If the suspend action is performed, the value changes to suspended. After the resume action is performed, the value changes back to empty.";

triggerEventAcceptorBehaviour BEHAVIOUR

DEFINED AS "The launch pad has attributes to monitor a specific attribute in a specific object instance. If the value of the monitored attribute is changed, a trigger to launch the scripts specified by the script ids specified by the default execution parameter list attribute is generated. In the case that the monitored attribute is a counter of EDC (Event Discrimination Counter) defined in Annex C, the notifications through the EDC trigger the launching of script execution by the launch pad.";

triggerActionAcceptorBehaviour BEHAVIOUR

DEFINED AS "When an instance of this class which is on duty receives a trigger, from a trigger activator, if its scriptId attribute is not empty, a new object is created in which the script id and class of the new instance come from the value of this instance's scriptId attribute and any of its other attributes and any parameters carried by the trigger.";

threadBehaviour BEHAVIOUR

DEFINED AS "When an instance of an object of this class is created, it begins execution of the command sequence specified through its attributes, using its parameter list to supply any parameters needed by the script. When execution of this sequence is complete, the object is deleted. If execution of the script causes the creation of contained threads, this thread is not considered complete until all contained threads are complete.";

parameterPasserBehaviour BEHAVIOUR

DEFINED AS "An instance of an object of this class passes a set of parameters to an instance of an object of another class.";

simpleScriptExecutionBehaviour BEHAVIOUR

DEFINED AS "A script is executed or interpreted by local means. Its execution status mirrors the following states:

- activated (spontaneous transition to next state: executing);
- executing (next state: timed out or completed);
- timed out (spontaneous transition to next state: completed);
- completed.

NOTE - Timeout value is implementation dependent.";

generalStringScriptBehaviour BEHAVIOUR

DEFINED AS "The syntax and semantics of scripting language which can be represented as general string. See Annexes F and G for details on one such language, SMSL.";

## A.4 Attribute definitions

availableScriptList ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.AvailableScriptList;

MATCHES FOR EQUALITY;

BEHAVIOUR

availableScriptListBehaviour BEHAVIOUR

DEFINED AS "A set of managed object instance names of the script instructions which can be executed by a launch pad.";

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx1(1)};

commandSequencerId ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.CommandSequencerId;

MATCHES FOR EQUALITY;

BEHAVIOUR

```

commandSequencerIdBehaviour    BEHAVIOUR
    DEFINED AS "The managed object instance name of the command sequencer.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx2(2)};

executionResultType ATTRIBUTE
    WITH ATTRIBUTE SYNTAX CSMModule.ExecutionResultType;
    MATCHES FOR EQUALITY;
    BEHAVIOUR
        executionResultTypeBehaviour BEHAVIOUR
            DEFINED AS "This indicates the type of execution result.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx3(3)};

scriptContent ATTRIBUTE
    WITH ATTRIBUTE SYNTAX CSMModule.ScriptContent;
    MATCHES FOR EQUALITY;
    BEHAVIOUR
        scriptContentBehaviour BEHAVIOUR
            DEFINED AS "The contents of a launch script represented by a general string.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx4(4)};

scriptId    ATTRIBUTE
    WITH ATTRIBUTE SYNTAX CSMModule.ScriptId;
    MATCHES FOR EQUALITY;
    BEHAVIOUR
        scriptIdBehaviour    BEHAVIOUR
            DEFINED AS "The managed object instance name of the script to be executed.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx5(5)};

launchPadId ATTRIBUTE
    WITH ATTRIBUTE SYNTAX CSMModule.LaunchPadId;
    MATCHES FOR EQUALITY;
    BEHAVIOUR
        launchPadIdBehaviour BEHAVIOUR
            DEFINED AS "The managed object instance name of the launch pad.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx6(6)};

scriptLanguageName ATTRIBUTE
    WITH ATTRIBUTE SYNTAX CSMModule.ScriptLanguageName;
    MATCHES FOR EQUALITY;
    BEHAVIOUR
        scriptLanguageNameBehaviour BEHAVIOUR
            DEFINED AS "The managed object instance name of a launch script represented by a
            general string.>";
REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx7(7)};

```

defaultExecutionParameterList ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.ExecutionParameterList;

MATCHES FOR EQUALITY;

BEHAVIOUR

defaultExecutionParameterListBehaviour BEHAVIOUR

DEFINED AS "A set of managed object instance names of the script instructions and parameter values (if required) as inputs to instances to be executed by default.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx8(8)};

executingParameters ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.ExecutionParameter;

MATCHES FOR EQUALITY;

BEHAVIOUR

executingParametersBehaviour BEHAVIOUR

DEFINED AS "A set of managed object instance names of the script instructions and parameter values (if required) as inputs to script executions.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx9(9)};

threadId ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.ThreadId;

MATCHES FOR EQUALITY;

BEHAVIOUR

threadIdBehaviour BEHAVIOUR

DEFINED AS "The managed object instance name of a thread executing script instruction(s).";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx10(10)};

triggerId ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.TriggerId;

MATCHES FOR EQUALITY;

BEHAVIOUR

triggerIdBehaviour BEHAVIOUR

DEFINED AS "The managed object instance name of a trigger initiating the execution of a launch script.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx11(11)};

## A.5 Notification definitions

executionResultInfo NOTIFICATION

WITH INFORMATION SYNTAX CSMModule.ExecutionResultInfo;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) notification(10) xx1(1)};

triggerResultInfo NOTIFICATION

WITH INFORMATION SYNTAX CSMModule.TriggerResultInfo;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) notification(10) xx2(2)};

**A.6 Action definitions****resume ACTION**

BEHAVIOUR resumeBehaviour BEHAVIOUR

DEFINED AS "An action directed at a basicSpawnerClass object, causing unconditional resumption of all script executions by the basicSpawnerClass object which were initiated by a particular trigger. The value of controlStatus becomes empty as the result of a resume action.";;

WITH INFORMATION SYNTAX CSModule.SpawnerObjectId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx1(1)};

**suspend ACTION**

BEHAVIOUR suspendBehaviour BEHAVIOUR

DEFINED AS "An action directed at a basicSpawnerClass object, causing unconditional suspension of all script executions by the basicSpawnerClass object which were initiated by a particular trigger. The value of controlStatus becomes 'suspended' as a result of a suspend action.";;

WITH INFORMATION SYNTAX CSModule.SpawnerObjectId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx2(2)};

**terminate ACTION**

BEHAVIOUR terminateBehaviour BEHAVIOUR

DEFINED AS "An action directed at a launch pad, causing unconditional termination of all scripts by the launch pad, which was initiated by a particular trigger.";;

WITH INFORMATION SYNTAX CSModule.TriggerId;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx3(3)};

**trigger ACTION**

BEHAVIOUR triggerBehaviour BEHAVIOUR

DEFINED AS "An initiator of script execution by causing a launch pad to spawn one or more threads.";;

WITH INFORMATION SYNTAX CSModule.TriggerParameters;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) action(9) xx4(4)};

**A.7 Name binding definitions****commandSequencer-system NAME BINDING**

SUBORDINATE OBJECT CLASS commandSequencer AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS system AND SUBCLASSES;

WITH ATTRIBUTE commandSequencerId;

BEHAVIOUR csSystemContainmentBehaviour BEHAVIOUR

DEFINED AS "Superior object class is system and subordinate object class is commandSequencer.";;

CREATE WITH-REFERENCE-OBJECT, WITH-AUTOMATIC-INSTANCE-NAMING;

DELETE ONLY-IF-NO-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx1(1)};

**launchPad-commandSequencer NAME BINDING**

SUBORDINATE OBJECT CLASS launchPad AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS commandSequencer AND SUBCLASSES;

WITH ATTRIBUTE launchPadId;

BEHAVIOUR lpCsContainmentBehaviour BEHAVIOUR

DEFINED AS "Naming a command sequence launch pad with respect to a command sequencer indicates that the command sequencer is the service provider for the launch pad.";;

CREATE WITH-AUTOMATIC-INSTANCE-NAMING;

DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx2(2)};

thread-synchronousLaunchPad NAME BINDING

SUBORDINATE OBJECT CLASS thread AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS synchronousLaunchPad AND SUBCLASSES;

WITH ATTRIBUTE threadId;

BEHAVIOUR threadSyncLpContainmentBehaviour BEHAVIOUR

DEFINED AS "The superior object class synchronousLaunchPad acts as an IVMO for the subordinate object class thread.";;

DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx3(3)};

suspendableThread-asynchronousLaunchPad NAME BINDING

SUBORDINATE OBJECT CLASS suspendableThread AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS asynchronousLaunchPad AND SUBCLASSES;

WITH ATTRIBUTE threadId;

BEHAVIOUR threadAsyncLpContainmentBehaviour BEHAVIOUR

DEFINED AS "The superior object class asynchronousLaunchPad acts as an IVMO for the subordinate object class suspendable thread.";;

DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx4(4)};

thread-thread NAME BINDING

SUBORDINATE OBJECT CLASS thread AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS thread AND SUBCLASSES;

WITH ATTRIBUTE threadId;

BEHAVIOUR threadContainmentBehaviour BEHAVIOUR

DEFINED AS "The superior object class thread acts as a spawner of the subordinate object class thread.";;

DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx5(5)};

suspendableThread-suspendableThread NAME BINDING

SUBORDINATE OBJECT CLASS suspendableThread AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS suspendableThread AND SUBCLASSES;

WITH ATTRIBUTE threadId;

BEHAVIOUR suspendableThreadContainmentBehaviour BEHAVIOUR

DEFINED AS "The superior object class suspendable thread acts as a spawner of the subordinate object class suspendable thread.";;

DELETE DELETES-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx6(6)};

launchScript-system NAME BINDING

SUBORDINATE OBJECT CLASS launchScript AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS system AND SUBCLASSES;

WITH ATTRIBUTE scriptId;

BEHAVIOUR lsSystemContainmentBehaviour BEHAVIOUR

DEFINED AS "The superior object class is system and subordinate object class is launchScript.";;

CREATE WITH-REFERENCE-OBJECT, WITH-AUTOMATIC-INSTANCE-NAMING;

DELETE ONLY-IF-NO-CONTAINED-OBJECTS;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) nameBinding(6) xx7(7)};

## A.8 ASN.1 definitions

CModule {joint-iso-itu-t ms(9) function(2) part21(21) asn1Module(2) 0}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS everything

IMPORTS

SimpleNameType

FROM Attribute-ASN1Module {joint-iso-itu-t ms(9) smi(3) part2(2)

asn1Module(2) 1 }

ObjectInstance, Attribute, CMISync, CMISFilter, ModifyOperator, Scope,

BaseManagedObjectId FROM CMIP-1 {joint-iso-itu-t ms(9) cmip(1) modules(0)

protocol(3)}

AE-title FROM ACSE-1 {joint-iso-itu-t association-control(2) abstract-syntax(1)

apdus(0) version(1)};

cmdSeqRelationshipClasses OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2) part21(21) relationshipClass(11)}

cmdSeqRelationshipMappings OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2) part21(21) relationshipMapping(12)}

cmdSeqRelationshipRoles OBJECT IDENTIFIER ::= {joint-iso-itu-t ms(9) function(2) part21(21) relationshipRole(13)}

--

-- Range Constraints used for relationship class definitions

--

RangeFromOneToOne ::= INTEGER (1 .. 1)

RangeFromZeroToMax ::= INTEGER (0 .. MAX)

--

```
-- Counter size constraint
```

```
--
```

```
MaxCounterSize ::= INTEGER{unlimited(0)}-- size in octets
```

```
ExecutionResultInfo ::= SEQUENCE {triggerId TriggerId,
                                   scriptId ScriptId,
                                   threadId ThreadId,
                                   errorCode ErrorCode,
                                   executionResultType ExecutionResultType,
                                   executionResult SET OF Attribute}
```

```
TriggerResultInfo ::= SEQUENCE {triggerId TriggerId,
                                CHOICE {singleTriggerResult ResultInfoFromThread,
                                         sequentialTriggerResult SEQUENCE OF
                                             ResultInfoFromThread,
                                         parallelTriggerResult SET OF ResultInfoFromThread}}
```

```
ResultInfoFromThread ::= SEQUENCE{executionType ExecutionType,
                                   errorCode ErrorCode,
                                   executionResultType ExecutionResultType,
                                   executionResult SET OF Attribute}
```

```
ExecutionType ::= CHOICE {singleExecution ScriptThreadSet,
                           parallelExecution SET OF ScriptThreadSet,
                           sequentialExecution SEQUENCE OF ScriptThreadSet}
```

```
ScriptThreadSet ::= SEQUENCE {scriptId ScriptId,
                              threadId ThreadId}
```

```
SpawnerObjectId ::= SEQUENCE {triggerId TriggerId,
                               CHOICE { threadId ThreadId,
                                         launchPadId LaunchPadId}}
```

```
ExecutionResultType ::= OBJECT IDENTIFIER
```

```
CommandSequencerId ::= ObjectInstance
```

```
ScriptId ::= ObjectInstance
```

```
ThreadId ::= ObjectInstance
```

```
TriggerId ::= ObjectInstance
```

```
LaunchPadId ::= ObjectInstance
```

```
ScriptList ::= CHOICE {scriptId ScriptId,
                       sequentialScriptList SEQUENCE OF ScriptId,
                       parallelScriptList SET OF ScriptId}
```

```
AvailableScriptList ::= SET OF ScriptList
```

```
emptyScriptList AvailableScriptList ::= {}
```

```
emptyExecutionParameterList ExecutionParameterList ::= sequentialExecutionList: {}
```

```
TriggerParameters ::= SEQUENCE {triggerId TriggerId,
                                 executionParameterList ExecutionParameterList}
```

```

ExecutionParameterList ::= CHOICE {executionParameter ExecutionParameter,
                                   sequentialExecutionList SEQUENCE OF
                                   ExecutionParameter,
                                   parallelExecutionList SET OF
                                   ExecutionParameter}

ExecutionParameter ::= SEQUENCE {scriptId ScriptId,
                                   scriptParameters SEQUENCE OF Attribute}

emptyParameterList   ExecutionParameterList ::= sequentialExecutionList:{ }

ErrorCode ::= SET OF INTEGER {noError(0),
                               noScriptError(1),
                               scriptRejectedError(2),
                               invalidParameterTypeError(3),
                               invalidParameterValueError(4),
                               scriptSyntaxError(5),
                               scriptExecutionFailedError(6),
                               invalidParameterNumber(7),
                               unauthorizedAccessError(8)}

ScriptLanguageName ::= OBJECT IDENTIFIER

ScriptContent ::= GeneralString

ModificationList ::= SET OF SEQUENCE{modifyOperator {2} IMPLICIT
                                   ModifyOperator DEFAULT replace,
                                   attributeId AttributeId,
                                   attributeValue ANY DEFINED BY
                                   attributeId OPTIONAL
                                   -- absent for setToDefault
                                   }

```

**END**

## Annex B

## General Relationship Model

(This annex forms an integral part of this Recommendation / International Standard)

This following is the GRM for the command sequencer.

--

-- *The following relationship classes support the command sequencer model*

--

commandSequencer-launchPadRelationshipClassBehaviour  
BEHAVIOUR DEFINED AS

!

The relationship class is concerned with the relationship between a command sequencer and its launch pads used to initiate the execution of scripts by means of threads requiring the support services provided by the command sequencer.

!;

commandSequencer-LaunchPad-bindingBehaviour

BEHAVIOUR DEFINED AS

!

This notification occurs upon the binding of a launch pad into the command sequencer / launch pad relationship.

!;

commandSequencer-LaunchPad-unbindingBehaviour

BEHAVIOUR DEFINED AS

!

This notification occurs when a launch pad is removed from the command sequencer / launch pad relationship. A launch pad may be removed from the relationship only if its administrative state is locked.

!;

commandSequencer-launchPad-RelationshipClass  
RELATIONSHIP CLASS

BEHAVIOUR commandSequencer-launchPadRelationshipClassBehaviour,

commandSequencer-LaunchPad-bindingBehaviour,

commandSequencer-LaunchPad-unbindingBehaviour;

SUPPORTS

ESTABLISH,

QUERY,

TERMINATE,

NOTIFY commandSequencer-LaunchPad-binding,

NOTIFY commandSequencer-LaunchPad-unbinding;

ROLE commandSequencerRole

COMPATIBLE-WITH commandSequencer

PERMITTED-ROLE-CARDINALITY CONSTRAINT CASN1.RangeFromOneToOne

REQUIRED-ROLE-CARDINALITY-CONSTRAINT CASN1.RangeFromOneToOne

PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

CASN1.RangeFromOneToOne

REGISTERED AS { CSMModule.cmdSeqRelationshipRoles 1 }

ROLE launchPadRole

COMPATIBLE-WITH launchPad

PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSMModule.RangeFromZeroToMax

REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSMModule.RangeFromZeroToMax

BIND-SUPPORT

UNBIND-SUPPORT

PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT CSMModule.RangeFromOneToOne

REGISTERED AS { CSMModule.cmdSeqRelationshipRoles 2 };

REGISTERED AS { CSMModule.cmdSeqRelationshipClasses 1 };

commandSequencer-LaunchPad-RelationshipMapping-Behaviour

BEHAVIOUR DEFINED AS

!

This relationship mapping describes how the command sequencer to launch pad relationship class may be represented using containment. In this relationship mapping, the command sequencer is the superior object for the purposes of naming, and launch pads are contained by it. Participation in this relationship implies that the launch pad and its spawn may use the services provided by the resource represented by the command sequencer.

!;

commandSequencer-launchPad-RelationshipMapping

RELATIONSHIP MAPPING

RELATIONSHIP CLASS

commandSequencer-launchPad-RelationshipClass;

BEHAVIOUR commandSequencer-launchPad-RelationshipMapping-Behaviour;

ROLE commandSequencerRole

RELATED-CLASSES commandSequencer  
REPRESENTED BY NAMING

launchPad-commandSequencer-NameBinding  
USING SUPERIOR,

ROLE launchPadRole

RELATED-CLASSES launchPad  
REPRESENTED BY NAMING

launchPad-commandSequencer-NameBinding  
USING SUBORDINATE;

OPERATIONS MAPPING

ESTABLISH MAPS-TO-OPERATION

CREATE commandSequencer OF commandSequencerRole

TERMINATE MAPS-TO-OPERATION

DELETE commandSequencer OF commandSequencerRole

```

NOTIFY commandSequencer-launchPad-binding
    MAPS-TO-OPERATION
        NOTIFICATION "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":
            objectCreationNotification
            OF commandSequencerRole
NOTIFY commandSequencer-launchPad-unbinding
    MAPS-TO-OPERATION
        NOTIFICATION "CCITT Rec. X.721 | ISO/IEC 10165-2:1992":
            objectDeletionNotification
            OF commandSequencerRole
BIND    MAPS-TO-OPERATION
        CREATE launchPad OF launchPadRole
UNBIND  MAPS-TO-OPERATION
        DELETE launchPad
        OF launchPadRole
QUERY  MAPS-TO-OPERATION
        GET OF commandSequencerRole
        GET OF launchPadRole;
REGISTERED AS {CSModule.cmdSeqRelationshipMappings 1}

```

scriptReferenceRelationshipClassBehaviour

BEHAVIOUR DEFINED AS

!

This relationship class describes the relationship existing between scripts and an object which references them for the purposes of identifying task(s) to be carried out.

!;

scriptReferencerRelationshipClass

RELATIONSHIP CLASS

BEHAVIOUR scriptReferencerRelationshipClassBehaviour;  
SUPPORTS

ESTABLISH,  
TERMINATE,  
QUERY;

ROLE scriptUserRole

COMPATIBLE-WITH scriptReferencer  
PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne  
REQUIRED-ROLE-CARDINALITY CONSTRAINT CSModule.RangeFromOneToOne  
PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT  
CSModule.RangeFromZeroToMax

REGISTERED AS {CSModule.cmdSeqRelationshipRoles 3}

ROLE scriptRole

COMPATIBLE-WITH launchScript  
PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne  
REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeFromOneToOne

PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

CModule.RangeFromZeroToMax

REGISTERED AS {CModule.cmdSeqRelationshipRoles 4}

REGISTERED AS {CModule.cmdSeqRelationshipClasses 2}

launchPad-launchScriptRelationshipMappingBehaviour BEHAVIOUR

DEFINED AS

!

This relationship mapping describes the relationship between the launch pad and the launch script. The launch pad initiates execution of the launch script and references it for the purposes of execution.

!;

launchPad-LaunchScriptMapping  
RELATIONSHIP MAPPING

RELATIONSHIP CLASS scriptReferenceRelationshipClass;

BEHAVIOUR launchPad-launchScriptMappingBehaviour;

ROLE scriptUserRole

RELATED CLASSES launchPad

REPRESENTED BY ATTRIBUTE scriptList

ROLE scriptRole

RELATED CLASSES launchScript;

OPERATIONS MAPPING

ESTABLISH MAPS-TO-OPERATION

CREATE launchPad OF scriptUserRole

-- using SET-BY-CREATE of scriptList --

REPLACE scriptList of scriptUserRole

-- which effectively adds a scriptId to scriptList --

ADD scriptList of scriptUserRole,

TERMINATE MAPS-TO-OPERATION

DELETE launchPad OF scriptUserRole

REPLACE scriptList OF scriptUserRole

-- which effectively removes a scriptId from scriptList --

REMOVE scriptList OF scriptUserRole,

QUERY MAPS-TO-OPERATION

GET scriptList OF scriptUserRole;

REGISTERED AS {CModule.cmdSeqRelationshipMappings 2};

thread-launchScriptRelationshipMappingBehaviour BEHAVIOUR

DEFINED AS

!

This relationship class describes the relationship between a thread and launch script. The launch script is referenced by the thread by means of the scriptId attribute from the scriptIds in the scriptList.

!;

thread-launchScriptMapping  
RELATIONSHIP MAPPING

```

RELATIONSHIP CLASS scriptReferenceRelationshipClass;

BEHAVIOUR thread-launchScriptRelationshipMappingBehaviour;

ROLE scriptUserRole

```

```

    RELATED CLASSES thread

```

```

    REPRESENTED BY ATTRIBUTE scriptId

```

```

    QUALIFIED BY scriptList

```

```

ROLE      scriptRole

```

```

    RELATED CLASSES launchScript;

```

```

OPERATIONS MAPPING

```

```

    ESTABLISH MAPS-TO-OPERATION

```

```

        CREATE OF scriptUserRole,

```

```

    TERMINATE MAPS-TO-OPERATION

```

```

        DELETE OF scriptUserRole,

```

```

    QUERY MAPS-TO-OPERATION

```

```

        GET scriptId OF scriptUserRole;

```

```

REGISTERED AS {CSModule.cmdSeqRelationshipMappings 3};

```

```

spawnner-progeny-RelationshipClass

```

```

RELATIONSHIP CLASS

```

```

    BEHAVIOUR spawnner-progenyRelationshipClassBehaviour BEHAVIOUR

```

```

DEFINED AS

```

```

    !

```

When an instance of this class causes the creation of new objects, it serves as an IVMO during the creation by supplying values for the new object's attributes based on its own defaultExecutionParameterList attribute and any parameters which were supplied to it as part of the action or local Instances of this class are capable of causing the creation of new object instances. The newly created instances will be contained by this instance, and their names will be automatically generated. Until all created objects are complete, this object's usage status will be "in use". If this object's administrative state is "locked" such new objects cannot be created. If, due to local resource limitations, this object is incapable of supporting more contained objects, its usage status will be "busy".

An instance of this class may cause the creation of new object instances from a single scriptId, from a set of scriptIds in any order or from a sequence of scriptIds in the order specified in the list i.e. after the first has been created the second may not be created until after the first is completed, and so on. The value of the created object's scriptId attribute gets its value from the corresponding element of this object's scriptList.

```

    !;

```

```

SUPPORTS

```

```

    ESTABLISH

```

```

    TERMINATE;

```

```

ROLE spawnnerRole COMPATIBLE-WITH basicSpawnerClass

```

```

    PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeOneToOne

```

```

    REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeOneToOne

```

PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT

CSModule.RangeOneToOne

REGISTERED AS {CSModule.cmdSeqRoles 5}

ROLE progenyRole COMPATIBLE-WITH thread

PERMITTED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeZeroToMax

REQUIRED-ROLE-CARDINALITY-CONSTRAINT CSModule.RangeZeroToMax

PERMITTED-RELATIONSHIP-CARDINALITY-CONSTRAINT  
CSModule.RangeOneToOne

REGISTERED AS {CSModule.cmdSeqRoles 6}

REGISTERED AS {CSModule.cmdSeqRelationshipClasses 3};

## Annex C

**Management Information Definitions for Event Discrimination Counting**

(This annex does not form an integral part of this Recommendation | International Standard)

The Event Discrimination Counter (EDC) class object counts the number of input events. Before counting up, the EDC tests values of attributes related to the event or an object generating the event and discriminates the event.

**C.1 Managed object class**

eventDiscriminationCounter MANAGED OBJECT CLASS

DERIVED FROM "DMI":discriminator;

CHARACTERIZED BY

edcPackage PACKAGE

BEHAVIOUR edcBehaviour BEHAVIOUR

DEFINED AS

"If the result of discrimination of a potential event report evaluates to TRUE and the event discrimination counter is in the Unlocked and Enabled state and does not exhibit the off-duty availability status, then the counter value of the counter attribute is incremented.";;

ATTRIBUTES

"CCITT Rec. 721 | ISO/IEC 10165-2:1992":counter GET,  
maxCounterSize GET;

NOTIFICATIONS

"CCITT Rec. 721 | ISO/IEC10162:1992":processingErrorAlarm;;;

CONDITIONAL PACKAGES

counterAlarmPackage PRESENT IF "a counter is of finite size and a notification is triggered by a capacity alarm threshold.";

REGISTERED AS {joint-iso-itu-t ms(9) ms(9) function(2) part21(21)

managedObjectClass(3) xx11(11));

**C.2 Package**

counterAlarmPackage PACKAGE

BEHAVIOUR

counterAlarmBehaviour BEHAVIOUR

DEFINED AS "When the counter value reaches the capacity alarm threshold(as a percentage of maximum counter size), the EDC(Event Discrimination Counter) generates an event indicating that a capacity threshold has been reached or exceeded. In reporting the capacity threshold event, use is made of the alarm report defined in CCITT Rec. X.733 | ISO/IEC 10164-4. Only the following parameters of the alarm report shall be used and all parameters are mandatory when used for reporting counter capacity threshold alarms.

Managed Object Class - This parameter shall identify the counter class.

Managed Object Instance - This parameter shall identify the instance of the counter that generated the event.

Alarm Type - This parameter shall indicate that a processing error alarm has occurred.

Event time - This parameter carries the time at

which the capacity threshold event occurred.

Perceived Severity - This parameter will indicate the severity assigned to the capacity threshold event. When the 100% counter full condition is reached, a severity value of critical shall be assigned to this event.

Monitored Attributes - This parameter shall carry the maximum counter size attribute of the EDC.

Probable Cause - This parameter shall carry the value congestion.

Threshold Info - This parameter shall carry the capacity threshold value (as percentage of total capacity) that was reached or exceeded in generating this event.";;

#### ATTRIBUTES

"CCITT Rec. 721| ISO/IEC 10165-2:1992":capacityAlarmThreshold GET-REPLACE  
ADD-REMOVE;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) package(4) xx15(15)};

### C.3 Attribute

maxCounterSize ATTRIBUTE

WITH ATTRIBUTE SYNTAX CSMModule.MaxCounterSize;

MATCHES FOR EQUALITY, ORDERING;

BEHAVIOUR

maxSizeOrderingBehaviour BEHAVIOUR

DEFINED AS "This Attribute represent the largest value of the counter. The ordering in the same as for sequentially increasing positive integers except that a value of zero is largest and denotes infinite size.";;

REGISTERED AS {joint-iso-itu-t ms(9) function(2) part21(21) attribute(7) xx12(12)};