

CTA Standard

Task Model Description (CE TASK 1.0)

CTA-2018 R2016

(Formerly ANSI/CEA-2018)

March 2008

NOTICE

Consumer Technology Association (CTA)™ Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of the Consumer Technology Association from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than Consumer Technology Association members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by the Consumer Technology Association in accordance with the American National Standards Institute (ANSI) patent policy. By such action, the Consumer Technology Association does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This document does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this document to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

This document is copyrighted by the Consumer Technology Association (CTA)™ and may not be reproduced, in whole or part, without written permission. Federal copyright law prohibits unauthorized reproduction of this document by any means. Organizations may obtain permission to reproduce a limited number of copies by entering into a license agreement. Requests to reproduce text, data, charts, figures or other material should be made to the Consumer Technology Association (CTA)™.

(Formulated under the cognizance of the CTA **R7 Home Networks Committee.**)

Published by
©CONSUMER TECHNOLOGY ASSOCIATION 2016
Technology & Standards Department
www.cta.tech

All rights reserved

FOREWORD

This standard was developed under the auspices of the Consumer Electronics Association (CEA) R7 Home Network Committee.

All of the RelaxNG schemas and XML examples in this document have been automatically inserted from files and mechanically checked for syntactic validity.

First time readers are advised to first read the informative introduction in 6.

Contents

Page

1	SCOPE.....	1
2	CONFORMANCE (Informative)	1
2.1	Task Model Description.....	2
2.2	Task-Based Application	2
2.3	Grounding.....	2
3	REFERENCES.....	2
3.1	Other Standards Used	2
3.1.1	RelaxNG	2
3.1.2	ECMAScript	2
3.2	Normative References	2
3.3	Informative References	3
3.4	Reference Acquisition	3
4	TERM USAGE	4
4.1	Applications.....	4
4.2	Use of Shall, Should and May	4
4.3	Schemas	4
4.4	Namespaces	5
4.5	Classes and Instances.....	5
4.6	Functions	5
5	DEFINITIONS	5
6	INTRODUCTION (Informative).....	6
6.1	Tasks	6
6.2	Task Decomposition	7
6.3	Task Classes and Instances	7
6.4	Task-Based Applications	8
6.5	Task Engines	8
6.6	Task Generation and Recognition.....	9
6.7	Task Modeling	10
7	TASK MODEL DESCRIPTION.....	10
7.1	MIME Type	10
7.2	Format and Encoding	11
7.3	XML Namespace.....	11
7.4	Root Element	11
8	TASK.....	12
8.1	Input and Output Slots	13
8.1.1	Datatypes	14
8.1.2	Predefined Slots.....	14
8.2	User Intent Concepts	15
8.2.1	Predefined Semantic Roles.....	16
8.3	ECMAScript Task Instances	17
8.4	Precondition	17
8.5	Postcondition	17
8.5.1	The 'sufficient' Attribute	18
8.6	Side Effects.....	18

9	TASK DECOMPOSITION	19
9.1	Step Order	21
9.2	Skipping and Repeating Steps	21
9.3	Applicability Condition	22
9.4	Bindings	22
10	GROUNDING	24
10.1	Scripts.....	24
10.2	Grounding Queries	26
10.2.1	The ‘devices’ Task	27
10.2.2	The ‘user’ Task	27
10.2.3	The ‘about’ Task	27
10.2.4	The ‘task’ Task	28
11	RECOMMENDATIONS FOR APPLICATIONS	28
11.1	Unknown Elements and Attributes	28
11.2	Task Model Ordering	28
11.3	Order of Step Execution	28
11.4	User Intent Concepts	28
11.5	Preconditions	28
11.6	Postconditions	28
11.7	Non-primitive Datatypes	29
11.8	Applicability Conditions	29
11.9	Skipping and Repeating Steps	29
11.10	Script Execution	29
11.10.1	Predefined ‘\$execute’ Function	29
11.10.2	Predefined ‘\$getModel’ Function	30
11.10.3	Predefined ‘\$occurred’ Function	30
11.11	Grounding Queries	30
	Annex A (Normative) Complete RelaxNG Schema for Task Model Description	31
	Annex B (Informative) Example Task Model	35
	Annex C (Normative) UPnP Devices Exposing a Task Model Description	38
	Annex D (Normative) URC Targets Exposing a Task Model Description	39

Figures

Figure 1: Conformance to This Standard	1
Figure 2: Task Terminology	7
Figure 3: Expected Architecture for Using CEA-2018	8
Figure 4: Possible Refinement of Figure 3 with Generic Task Engine	9
Figure 5: Dataflow Diagram for BC Decomposition	24
Figure 6: Decomposition for playMusic Task	35

Tables

Table 1: RelaxNG Schema for Task Model 11

Table 2: RelaxNG Schema for Task..... 12

Table 3: RelaxNG Schema for Input and Output..... 13

Table 4: RelaxNG Schema for User Intent Concepts 15

Table 5: RelaxNG Schema for Subtasks..... 19

Table 6: RelaxNG Schema for Step Content 20

Table 7: RelaxNG Schema for Scripts..... 25

Table 8: Predefined Task Model for Grounding Queries 27

Table 9: RelaxNG Schema for About Document..... 27

CEA-2018 Task Model Description CE TASK 1.0

1 SCOPE

A task model is a formal description of the activities involved in completing a task, including both activities carried out by humans and those performed by machines. This standard defines the semantics and an XML notation for task models relevant to consumer electronics devices. The standard does not depend on any specific home networking technology or infrastructure.

2 CONFORMANCE (Informative)

Figure 1 illustrates the conformance approach of this standard. At the center of the diagram is a *task model description*, which is an XML document. A task model description is used by a *task-based application* to guide its interaction with a user. A task model description specifies task classes, and representations of their intent and how high-level tasks can be decomposed into lower-level tasks. A task model description also contains ECMAScript programs which ground primitive tasks to devices via particular networking platforms. This standard defines an XML language for task model descriptions which is independent of task-based applications. See 6 for a further informative introduction tasks and task modeling.

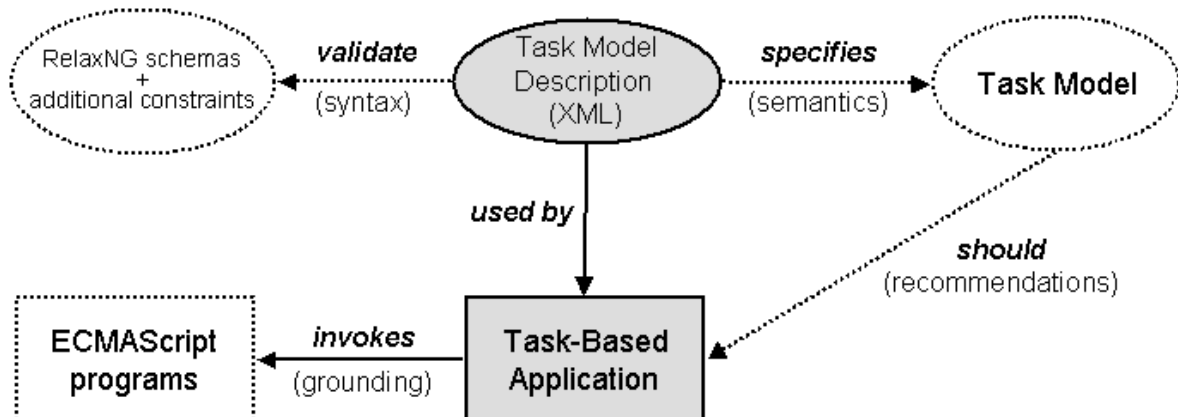


Figure 1: Conformance to This Standard

In general, the process of writing (authoring), distributing and using a task model description involves various stakeholders.

There are many different options for writing and distributing a task model description. First, a device manufacturer may write a task model description for their device, and either build it into the device so that it can be retrieved by a task-based application (see [REF 10.2.4]) or distribute it through their web server. Second, the developer of a task-based application may build task model descriptions into their application, or more likely, provide them through their web server. Finally, a third party may write task model descriptions for any device and make it available publicly through a web server.

Task model descriptions from different sources may mutually reference each other. For example, some task model descriptions may serve as "libraries" for specific domains. Similarly, the author of the ECMAScript program used to ground a particular task class (for a particular platform) may be different from the author of the task class specification.

2.1 Task Model Description

Conformance of task model descriptions is relevant to authors of task model descriptions, because it allows a task model description to be used by any task-based application based on this standard.

An XML document is a valid task model description if it conforms to the requirements outlined in 7 through 10 of this standard.

The syntactic validity of task model descriptions is completely and unambiguously established by the RelaxNG schemas in this standard, including the comments added to express restrictions that are not expressible in the schema language. (See Annex A for complete schemas.)

The semantics of task model descriptions is established by describing in words how a given document *specifies* a task model. A *task model* is an abstract *mathematical* construct, which is defined in this standard using common notions such as class, instance, set, function, etc.

2.2 Task-Based Application

Conformance of task-based applications is relevant to implementers of task-based applications, because that want their applications to make proper use of conformant task model descriptions.

The behavioral connection between a task-based application and the task model description it uses is established indirectly via the *recommendations* in 11. These are only recommendations, rather than mandatory requirements, because of the great variation possible in task-based applications. For example, this standard does not prevent an application from ignoring some aspect of the task model under direct instructions from the user.

2.3 Grounding

Conformance of the ECMAScript programs used for grounding is relevant for implementers of network platforms and devices, because it allows their platforms and devices to be used by task-based applications. The ECMAScript programs conform to this standard if they obey the specifications in 10.

3 REFERENCES

3.1 Other Standards Used

3.1.1 RelaxNG

The RelaxNG [6] schema language is used for specifying the XML syntax of task model descriptions.

NOTE See [14] for a short introductory tutorial on RelaxNG.

3.1.2 ECMAScript

ECMAScript [5] (commonly known as JavaScript) is used for grounding task model descriptions.

3.2 Normative References

The following standards contain provisions that, through reference in this text, constitute normative provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed.

- [1] Extensible Markup Language (XML) 1.0 (Fourth Edition) – W3C Recommendation 16 August 2006, edited in place 29 September 2006, <http://www.w3.org/TR/2006/REC-xml-20060816/>
- [2] IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, Nov. 1996, <http://www.ietf.org/rfc/rfc2046.txt>
- [3] IETF RFC 3986, Uniform Resource Identifier (URI): Generic Syntax, January 2005, <http://www.ietf.org/rfc/rfc3986.txt>

- [4] ISO/IEC 10646:2003, Universal Multiple-Octet Coded Character Set (UCS)
- [5] ISO/IEC 16262:2002, Information technology - ECMAScript language specification (also available from ECMA as Standard ECMA-262, ECMAScript Language Specification, 3rd edition, Dec. 1999)
- [6] ISO/IEC 19757-2:2003, Information technology – Document Schema Definition Language (DSDL) – Part 2: Regular-grammar-based validation – RELAX NG.
- [7] Namespaces in XML – W3C Recommendation 14 January 1999, <http://www.w3.org/TR/1999/REC-xml-names-19990114/>

3.3 Informative References

- [8] Dublin Core Metadata Element Set, Version 1.1, 2006-12-18, <http://www.dublincore.org/documents/2006/12/18/dces/>
- [9] DCMI Metadata Terms, 2003-11-19, <http://dublincore.org/documents/2003/11/19/dcmi-terms/>
- [10] Harley, Heidi. Thematic Roles. In: Patrick Hogan (ed.), The Cambridge Encyclopedia of Linguistics. Cambridge University Press, 2007.
- [11] ISO/IEC FDIS 24752-1: Information Technology – User Interfaces – Universal Remote Console — Part 1: Framework. To be published in 2008
- [12] ISO/IEC FDIS 24752-4: Information Technology – User Interfaces – Universal Remote Console — Part 4: Target Description. To be published in 2008
- [13] Merriam-Webster Online, 2007. <http://www.m-w.com/dictionary/task>
- [14] OWL Web Ontology Language Overview. W3C Recommendation 10 Feb 2004. <http://www.w3.org/TR/2004/REC-owl-features-20040210/>
- [15] RELAX NG Compact Syntax Tutorial, Working Draft 26 March 2003. OASIS, 2003. <http://www.relaxng.org/compact-tutorial-20030326.html>
- [16] “Thematic roles”. In: Santorini, Beatrice, and Anthony Kroch, The syntax of natural language: An online introduction using the Trees program, 2007. <http://www.ling.upenn.edu/~beatrice/syntax-textbook/box-thematic.html>
- [17] UPnP™ Device Architecture 1.0, version 1.0.1, 2 December 2003, UPnP forum, <http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf>
- [18] “Working Through Task-Centered System Design”, S. Greenberg, in *The Handbook of Task Analysis for Human-Computer Interaction*, D. Diaper and N. Stanton (eds), Lawrence Erlbaum, 2004, pp. 49-65. (See <http://grouplab.cpsc.ucalgary.ca/papers/2002/02-TaskAnalysis.Chapter/task-analysis.chapter.pdf>)

3.4 Reference Acquisition

ANSI/CEA Standards:

Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Phone 800-854-7179; Fax 303-397-2740; Internet <http://global.ihs.com>; E-mail global@ihs.com

DCMI Documents:

Dublin Core Metadata Initiative (DCMI); Internet <http://www.dublincore.org>

ECMA Documents:

ECMA International, Rue du Rhone 114, CH-1204 Geneva; Phone +41 22 849 6000; Fax +41 22 489 6001; Internet <http://www.ecma-international.org/>. ECMA Standard Documents can be downloaded electronically and free of charge at <http://www.ecma-international.org/publications/standards/Standard.htm>

IETF Documents:

Internet Engineering Task Force (IETF) Secretariat, c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 20191-5434, USA; Phone 703- 620-8990; Fax 703- 620-9071; Internet www.ietf.org, IETF RFCs may be downloaded from www.ietf.org/rfc.html, IETF Internet drafts may be downloaded from www.ietf.org/ID.html

ISO/IEC Documents:

International Organization for Standardization (ISO); ISO Central Secretariat; 1, rue de Varembe; Case postale 56; CH-1211 Geneva 20; Switzerland; Telephone +41 22 749 01 11; Fax +41 22 733 34 30; Internet <http://www.iso.org>; ISO/IEC Standard Documents can be purchased and downloaded electronically at <http://www.iso.org/iso/en/prods-services/ISOstore/store.html>

UPnP Documents:

UPnP Forum; Internet <http://www.upnp.org/>

World Wide Web Consortium Documents:

World Wide Web Consortium (W3C); Internet <http://www.w3.org/>

4 TERM USAGE

Notwithstanding the placement of a NOTE or EXAMPLE in a normative section, paragraphs marked as “NOTE” or “EXAMPLE” shall be treated as informative.

4.1 Applications

DEFINITION 1 A **task-based application** (“application” for short) is a program that interprets task model descriptions according to the semantics defined in this standard.

For brevity, in the remainder of this document, a task-based application will be referred to generically as an *application*.

4.2 Use of Shall, Should and May

Specific keywords are used in this document to differentiate levels of requirements and optionality, as follows:

Shall: Indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this standard.

Should: Indicates flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

May: Indicates flexibility of choice with no implied preference.

4.3 Schemas

The valid syntax for task model descriptions is defined using RelaxNG [6], with comments added to express restrictions that are not expressible in this schema language. These schemas are presented in Table 1 through Table 7 and are normative in this standard.

The syntax for the ‘about’ document (see 10.2.3) is also defined using RelaxNG. The corresponding Table 9 is normative in this standard.

Notice that the order of XML sibling elements is fixed throughout the syntax schemas (attributes may be provided in any order). The motivation for this approach is to make it easier for humans to read task model descriptions and for applications to process them.

4.4 Namespaces

This standard defines and uses the XML namespaces "http://ce.org/cea-2018" for task model descriptions. Examples in this standard assume that this namespace is declared as the default namespace on the root element of a task model description document, such as in the following:

```
<taskModel xmlns="http://ce.org/cea-2018" ...>
```

Task model descriptions may adopt this approach, or define an explicit namespace prefix. For other namespaces, the following three prefixes are used in the examples throughout this standard. Task model descriptions are not required to use these same prefixes as long as they declare the relevant namespaces properly (as specified in [7]).

- *xsd* -- the namespace <http://www.w3.org/2001/XMLSchema> (XML Schema Definition)
- *dc* -- the namespace <http://purl.org/dc/elements/1.1> (DCMI Element Set). A schema definition file for this namespace is available at <http://www.dublincore.org/schemas/xmls/qdc/2006/01/06/dc.xsd>
- *dcterms* -- the namespace <http://purl.org/dc/terms> (DCMI Metadata Terms). A schema definition file for this namespace is available at <http://www.dublincore.org/schemas/xmls/qdc/2006/01/06/dcterms.xsd>

Namespace URIs, such as those above, are primarily for identification purposes and are not required to be resolvable [7], i.e., they might not serve a document when used as an internet address.

4.5 Classes and Instances

The standard terminology of object-oriented programming and design is used in this document without formal definition. In particular, a *class* typically has many *instances*, which differ in the values of the instance variables, or *slots* (as they are called in this document).

DEFINITION 2 A **slot** is a named, typed instance variable.

A class determines the *names* of the slots and the *type* of values each slot may hold; the slot *values* are determined by each instance. Finally, the class itself may have *properties* (sometimes called class variables), which are "inherited" by all instances, i.e., all instances have the same value for these properties.

4.6 Functions

DEFINITION 3 A **partial function** is a function which may be undefined for some argument values.

DEFINITION 4 An undefined **function** is a partial function which is undefined for all argument values.

5 DEFINITIONS

For the purposes of this document, the following terms and definitions apply. (This section lists the definitions of this standard in one place.)

DEFINITION 1 A **task-based application** ("application" for short) is a program that interprets task model descriptions according to the semantics defined in this standard.

DEFINITION 2 A **slot** is a named, typed instance variable.

DEFINITION 3 A **partial function** is a function which may be undefined for some argument values.

DEFINITION 4 An **undefined function** is a partial function which is undefined for all argument values.

DEFINITION 5 A **task model** is an ordered set of task classes, decomposition classes and scripts. The empty set is a valid task model.

DEFINITION 6 A **task class** is a class whose properties include a precondition, a postcondition, a (possibly empty) ordered set of user intent concepts and a (possibly empty) set of scripts.

DEFINITION 7 A **task occurrence** is a task instance which corresponds to an actual event.

DEFINITION 8 A **hypothetical task instance** is a task instance which does not correspond to an actual event.

DEFINITION 9 An **input slot** is a slot of a task class which is expected to have a value *before* execution of a task instance.

DEFINITION 10 An **output slot** is a slot of a task class which is not expected to have a value until *after* execution of a task instance.

DEFINITION 11 A **user intent concept** is a case frame, consisting of a verb and a set of semantic roles of specified types.

DEFINITION 12 A **precondition** is a partial boolean function property of a task class with arguments corresponding to the input slots of the task class.

DEFINITION 13 A **postcondition** is a partial boolean function property of a task class with arguments corresponding to the input and output slots of the task class.

DEFINITION 14 A **decomposition class** is a class whose properties include a step order, applicability condition and binding set.

DEFINITION 15 An **applicability condition** is a partial boolean function property of a decomposition class or a script with arguments corresponding to the input slots of the goal or script task class.

DEFINITION 16 A **binding set** is a (possibly empty) set property of a decomposition class containing bindings.

DEFINITION 17 A **binding** is an equality between an input slot of a decomposition step or an output slot of a decomposition goal, and the value of a function with arguments corresponding either to the output slots of steps or the input slot of the goal.

DEFINITION 18 A **script** is an ECMAScript [5] program which may be associated with one or more tasks classes, platforms and device types and whose properties include an applicability condition.

DEFINITION 19 An **initialization script** is a script which is not associated with any task class and is intended to be executed exactly once.

6 INTRODUCTION (Informative)

6.1 Tasks

The concept of task is at the heart of this standard. A dictionary [10] definition of task is:

“a usually assigned piece of work often to be finished within a certain time.”

The task concept is also suggested by synonyms like "activity", "goal", "job", or "action". Examples of tasks in the domain of consumer electronics include copying a videotape to a DVD, watching a recorded TV episode, and turning off room lights. Tasks vary widely in their time extent: some take place over minutes or hours, e.g., watching a recorded TV episode; some are effectively instantaneous, e.g., turning off the room lights; and some have unbounded time extent, e.g., a weekly teleconference.

Tasks typically involve both human participants (e.g., as requesters, beneficiaries, or performers) and electronic devices. Some tasks may be performable only by a human being (e.g., providing fingerprint identification); others may be performed only by an electronic device (e.g., displaying a video); yet others may be performed by either, depending on the circumstances (e.g., opening the DVD drawer).

Tasks also vary along an abstraction spectrum from what might be called "high-level", i.e., closer to the user's intent and natural way of communicating, to "low-level", i.e., closer to the primitive controls of a particular device. Watching a recorded TV episode is an example of a fairly high-level task. Pressing the power button on a DVD player is an example of a very low-level task. Tasks are also more or less abstract by virtue of being parameterized. For example, watching the Season Two opening episode of *The West Wing* is a more specific version of the generic task of watching a recorded TV episode (in which the episode to be watched has been bound).

Specification of regularly scheduled tasks is out of the scope of this standard, since it can be very complicated.

6.2 Task Decomposition

High-level tasks usually need to be (repeatedly) decomposed into increasingly lower-level tasks (called *subtasks*) in order to accomplish them. This decomposition can sometimes be achieved entirely by the system; sometimes a collaboration between the system and user is required. The two main reasons why user involvement in task decomposition is sometimes required are:

- There are preferences involved which the user may need/want to provide. For example, when making a digital recording, a choice needs to be made in the fundamental tradeoff between picture quality and recording size. The system can help the user by explaining the options and their implications, and suggesting defaults. Similarly, the user may help in identifying which device should be used for the execution of a task, if multiple devices are available that can be used.
- The hardware may not be physically capable of performing certain actions, such as putting a blank DVD into the recorder.

6.3 Task Classes and Instances

A task model, as we will see below, contains task *classes*. For example, pressing the power button on a DVD player is an example of a task class. Slots of this class might include who pressed the button, which DVD player was involved, and when the action took place. Thus David Smith pressing the power button on the DVD player in his living room at 3:15pm on January 1, 2006 is an example of an *instance* of this class. A task instance may correspond to an actual event or it may be hypothetical. See 8.3 for the ECMAScript representation of task instances used in this standard.

Applications typically manipulate both task classes and instances. This standard does *not*, however, specify a representation for task instances – that is up to application implementations.

Figure 2 and the definitions in 8 summarize this terminology for tasks. For brevity, this standard sometimes simply uses the term *task* below where the context makes it clear whether a task class or a task instance is meant, or the remark applies to both classes and instances.

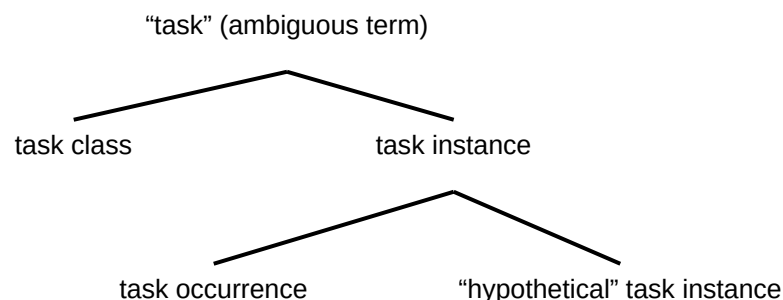


Figure 2: Task Terminology

6.4 Task-Based Applications

The motivation for defining this standard is to facilitate the development of task-based applications. Figure 3 shows the system architecture that this standard assumes. In a task-based application, the user and system interact primarily in terms of high-level goals, which the system decomposes into primitive actions that are directly supported by the one or more devices involved. Task-based applications address the problem that many people find new digitally-enhanced consumer electronic products too complex to operate and therefore do not use most of the features of them.

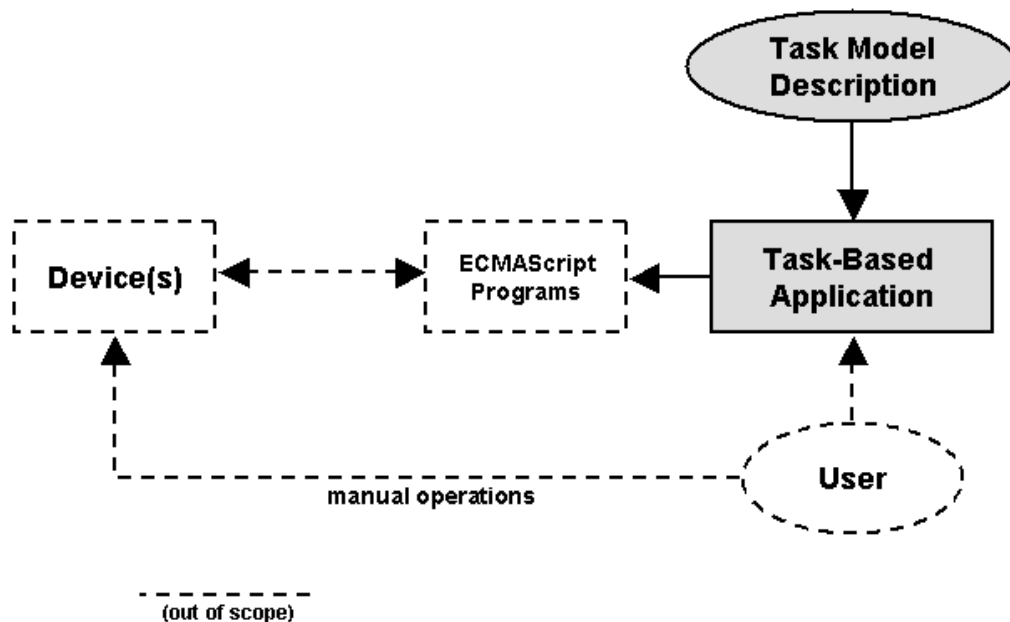


Figure 3: Expected Architecture for Using CEA-2018

Figure 3 is a functional, as opposed to physical, architecture. For example, the task-based application component may in fact run as a computational process on the same hardware as the device, or they may each have separate hardware. Furthermore, the display function of the user interface (if any) may share hardware with the device (e.g., if the device is a TV), or it may be a remote display.

Two fundamental inputs to the task-based application are the *task model description* (which is formalized in standard) and interaction with the user. Notice that the user interaction is out of the scope of this standard.

Communication between the task-based application and the device(s) is expected to be achieved via some kind of underlying network platform, such as UPnP. The framework for this communication is defined in 10 on grounding. The specific contents of this communication depend on the task models used.

Notice that this standard also allows for the possibility that the user will perform manual operations on the device, such as loading a CD. In fact, some devices may only be operable manually (e.g., they are not connected to the network), in which case the user interface would simply be providing instructions to the user for what to do. Manual operations are also out of the scope of this standard.

6.5 Task Engines

Figure 4 shows a possible refinement of Figure 3, in which the task-based application has been implemented in two parts: a generic *task engine* and an application-specific user interface which uses the task engine. Notice that all of this refinement is outside the scope of this standard.

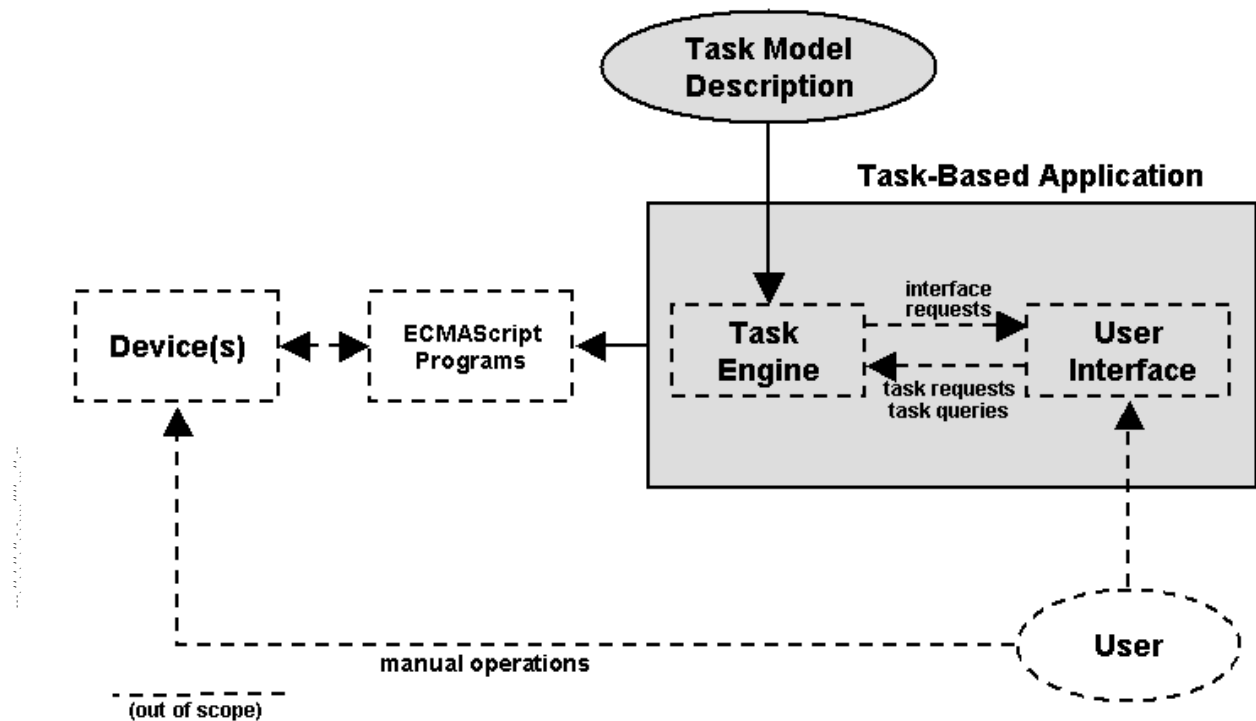


Figure 4: Possible Refinement of Figure 3 with Generic Task Engine

The basic functions of the task engine are to load the task model description and to maintain a representation of the current status of the user's task instances (see 6.3). The advantage of this approach is that the effort of building a task engine can be amortized over many different task-based applications. Notice that the task engine does not interact directly with the user. If the task engine needs input from the user, it sends an *interface request* to the user interface; how this request is presented to the user depends on the specific user interface, e.g., it may be visual, spoken, etc. Furthermore, the task engine provides an API through which the user interface may access the current task state (*task queries*) or cause a certain task to be performed (*task requests*).

Finally, like Figure 3, this is a functional architecture---the task engine and user interface may or may not be part of the same physical component, and the implementation of the communication between them depends on the situation.

6.6 Task Generation and Recognition

Two key functions typically performed by task-based applications are task generation and task recognition.

Task generation, sometimes called planning, is the process by which the system decides what it should do next. There are two different techniques typically used for this purpose, which may be intermixed. This standard is intended to support both techniques.

One task generation technique, called *first-principles planning*, searches through the set of all possible sequences of primitive actions to find a sequence which transitions from the current world state to the goal state. First-principles planning is particularly important for recovering from anticipated error conditions. First-principles planning requires a complete formalization of the preconditions and postconditions of each primitive action (see 8.4 and 8.5).